Санкт-Петербургский Государственный Университет
Кафедра компьютерного моделирования и многопроцессорных систем

Лысов Кирилл Александрович

Магистерская диссертация

Прогнозирование ценовых колебаний и долгосрочных трендов на финансовых рынках

Направление 02.04.02
Фундаментальная информатика и информационные технологии

Научный руководитель,
доктор физ.-мат. наук,
Академик Европейской академии Евросайенс и РАЕН, профессор
Богданов А. В.

Санкт-Петербург
2018

Оглавление

ВВЕДЕНИЕ 3
ПОСТАНОВКА ЗАДАЧИ 7
ОБЗОР ЛИТЕРАТУРЫ 8
ГЛАВА 1. Анализ методов глубинного обучения, библиотек и инструментов 14
1.1. Свёрточная нейронная сеть (CNN) 17
1.2. Рекуррентная нейронная сеть (RNN) 19
1.3. RNN: Sequence-to-Sequence 22
1.4. Автокодировщики 23
1.5. Обучение с подкреплением 25
1.6. Генеративно-состязательная сеть 27
1.7. Библиотеки и инструменты 28
ГЛАВА 2. Подготовка данных и построение предсказательной модели 30
2.1. Подготовка данных 30
2.2. Нормализация данных 34
2.3. Построение модели 34
ГЛАВА 3. Обучение, визуализация и результаты 38
3.1. Эксперименты 38
3.2. Визуализация 43
ВЫВОДЫ 46
ЗАКЛЮЧЕНИЕ 47
Список литературы и источников 48

ВВЕДЕНИЕ

Прогнозирование фондового рынка - это попытка определить будущую стоимость акций компании или другого финансового инструмента, торгуемого на бирже. Успешное прогнозирование будущей цены акций может принести значительную прибыль. Гипотеза об эффективном рынке предполагает, что цены на акции отражают всю имеющуюся в настоящее время информацию, и любые изменения цен, которые не основаны на недавно выявленной информации, по своей сути непредсказуемы. Другие не согласны, и те, у кого есть эта точка зрения, обладают бесчисленными методами и технологиями, которые предположительно позволяют им получать информацию о будущих ценах.

Методики прогнозирования делятся на три широкие категории, которые могут (и часто делают) перекрываться. Это *фундаментальный анализ*, *технический анализ* и *методы интеллектуального анализа данных*:

- **Фундаментальный анализ**

Сторонники фундаментального анализа обеспокоены компанией, которая лежит в основе самого индекса. Они оценивают прошлую работу компании, а также доверие к её финансовым показателям. Создаются многие коэффициенты производительности, которые помогают фундаментальному аналитику оценить стоимость акции.

Фундаментальный анализ основан на убеждении, что человеческое общество нуждается в капитале для достижения прогресса, и если

компания работает хорошо, она должна быть вознаграждена дополнительным капиталом и это приведет к резкому увеличению курса акций. Фундаментальный анализ широко используется менеджерами фондов, поскольку он является наиболее разумным, объективным и сделан из общедоступной информации, такой как анализ финансовой отчетности.

Другим значением фундаментального анализа является анализ «сверху вниз»: сначала анализируем глобальную экономику, затем страну, сектор экономики и, наконец, анализ конкретной компании.

- **Технический анализ**

Технические аналитики или чартисты (от англ. *chart* - график) не заинтересованы ни в одном из основных показателей компании. Они стремятся определить будущую цену акции, основываясь исключительно на (потенциальных) тенденциях прошлой цены (форме анализа временных рядов). Используются многочисленные шаблоны, такие как **«голова и плечи»** [1], **«чашка с ручкой»** [2]. Наряду с шаблонами используются методы, такие как **экспоненциальная скользящая средняя (EMA)** [3]. **«Японские свечи»** [4], которые, как полагают, были впервые разработаны японскими торговцами рисом, в наши дни широко используются техническими аналитиками.

- **Технологии интеллектуального анализа данных**

С распространением компьютера прогнозирование фондового рынка с тех пор перешло в технологическую сферу. Наиболее известный метод заключается в использовании **искусственных нейронных сетей (ANN)**

[5] и **генетических алгоритмов (GA)** [6]. Ученые обнаружили, что метод оптимизации бактериального хемотаксиса может работать лучше, чем GA [7]. ANN можно рассматривать как аппроксиматоры математической функции. Наиболее распространенной формой ANN, используемой для прогнозирования фондового рынка, является сеть прямой передачи, использующая алгоритм обратного распространения ошибок для обновления сетевых весов. Эти сети обычно называются сетями обратного распространения ошибки. Другой формой ANN, которая более подходит для прогнозирования цены, является **рекуррентная нейронная сеть (RNN)** [8] или **нейронная сеть с временной задержкой (TDNN)** [9]. Примерами RNN и TDNN являются сети Элмана, Джордана и Элмана-Джордана.

Для прогнозирования цены с помощью ANN обычно используются два подхода для прогнозирования разных временных горизонтов: независимые и совместные. В независимом подходе используется одна ANN для каждого временного горизонта, например, 1-дневный, 2-дневный или 5-дневный. Преимущество такого подхода заключается в том, что ошибка прогноза сети для одного горизонта не повлияет на ошибку для другого горизонта, поскольку каждый временной горизонт обычно является уникальной проблемой. Однако совместный подход включает в себя несколько временных горизонтов, чтобы они определялись одновременно. В этом подходе ошибка прогнозирования для одного временного горизонта может делить свою ошибку с ошибкой другого горизонта, что может снизить производительность. Для

совместной модели также требуется больше параметров, что увеличивает риск переобучения.

В последнее время большинство академических исследовательских групп, изучающих ANN для прогнозирования цен, по-видимому, с большим успехом используют ансамбль независимых методов ANN. Ансамбль ANN будет использовать низкие цены и временные задержки для прогнозирования будущих минимумов, в то время как другая сеть будет использовать текущие максимумы для прогнозирования будущих максимумов. Прогнозируемые низкие и высокие цены затем используются для формирования стоп-цен для покупки или продажи. Выходы из отдельных «низких» и «высоких» сетей также могут вводиться в конечную сеть, которая также будет включать объемы, межрыночные данные или статистические сводки цен, что приведет к окончательному выпуску ансамбля, который приведет к покупке, продаже или рыночному направлению изменение. Основной вывод с ANN и прогнозом запаса заключается в том, что подход классификации (аппроксимация функции) с использованием выходов в виде buy (y = + 1) и sell (y = -1) приводит к лучшей прогнозирующей надежности, чем количественный результат, такой как как низкая или высокая цена [10]. Это объясняется тем, что ANN может предсказать класс лучше, чем количественное значение, как при приближении функции, поскольку ANN иногда больше узнают о шуме во входных данных.

Поскольку ANN требуют обучения и могут иметь большое пространство параметров, полезно изменить структуру сети для оптимальной прогностической способности.

ПОСТАНОВКА ЗАДАЧИ

Прогнозирование фондового рынка связано не просто с индивидуальным извлечением прибыли конкретными игроками, но так же с общим увеличением эффективности и стабильности финансовой отрасли и экономики в целом путем увеличения ликвидности. Те участники, которые нуждаются в капитале и те, кто хочет свой капитал приумножить, получают больше возможностей для достижения своих целей.

Целью данной работы является построение эффективной и ресурсоемкой модели прогнозирования поведения финансовых инструментов, трендов и движения цен по принципам глубинного обучения.

Решение данной задачи декомпозируется на ряд подзадач:

- Анализ различных методов глубинного обучения, библиотек и инструментов, находящихся в открытом доступе, с целью применения;
- Поиск, предобработка и нормализация данных для обучения и тестирования;
- Программная реализация модели;
- Визуальное представление результатов.

На выходе мы должны получить программный комплекс для прогнозирования направления движения цен на финансовых рынках.

ОБЗОР ЛИТЕРАТУРЫ

При написании данной работы были использованы научные статьи, блоги, диссертации и другие материалы исследователей Университета Аризоны, Принстонского Университета, Калифорнийского университета в Беркли, Университета Западного Онтарио, Китайской Академии Наук, Национального Университета Сингапура, Университета Тегерана и пр.

Ученые Роберт П. Шумакер и Синьчунь Чен из Университета Аризоны провели текстовый анализ прогноза фондового рынка с использованием последних финансовых новостей [11]. В их исследовании рассматривается метод машинного обучения на финансовых новостях с использованием нескольких различных текстовых представлений: **Bag of Words** [19], **именных групп (ИГ)** [20] и **именованных сущностей** [21]. Они исследовали 9,216 статей финансовых новостей и 10,259,042 котировок акций, покрывающие S&P 500 в течение пяти недель. Получившаяся модель была использована для оценки дискретной цены акции через двадцать минут после публикации новостной статьи. Вместе с модифицированной версией **метода опорных векторов (SVM)** [22], специально разработанной для дискретного численного прогнозирования и моделями содержащими различные переменные финансовых индексов, ученые выяснили, что модель, содержащая анализ финансовых статей и цен акций на момент выпуска статьи имела наилучшую производительность вблизи к фактической будущей цене

акции (**MSE 0,04261**), в том же направлении движения цены, что и будущая цена (**57,1%** точности) и максимальной выгодой с использованием моделируемой торговой системы (**2,06%** рост прибыли). Ученые дополнительно исследовали различные текстовые представления и обнаружили, что грамотно составленная именная группа работает лучше, чем стандартная Bag of Words во всех трех метриках.

Радж Тхакур, Санчита Бадкас, Минакумари Паде и Паллави Худе. из Технического Колледжа Патила [12] построили гибридную систему на нейронных сетях, которая пытается предсказать цену открытия фондового рынка для нескольких конкретных компаний на основаннии прошлой деятельности компании и информации из различных новостных источников, объединяя элементы фундаментального и технического анализа и подтвержая жизнеспособность данного подхода.

Феликс Мин Фай Вонг, Чжэньминь Лю, Мунг Чанг из Принстона [13] построили систему, которая предсказывает движение цен только по новостям с Wall Street Journal с использованием **Text Mining** [23] и **факторизации разреженной матрицы** [24]. Они пересматривают проблему прогнозирования направленности движения цен на акции, основанной на новостных статьях: здесь алгоритм использует ежедневные статьи из The Wall Street Journal для прогнозировать цены закрытия акций в тот же день. Ученые предлагают унифицированную модель скрытого пространства для определения «совместных движений» между ценами акций и новостными статьями. В отличие от многих существующих подходов, новая модель способна одновременно использовать корреляции: (а) между ценами акций, (б) среди новостных

статей и (c) между ценами на акции и новостными статьями. Таким образом, модель способна ежедневно прогнозировать более 500 акций (большая часть которых даже не упоминаются ни в одной новостной статье) при низкой сложности. Ученые провели обширные опросы по торговым стратегиям, основанным на созданном алгоритме. Результат показывает, что модель имеет значительно лучшую точность (55,7%) относительно многих широко используемых алгоритмов. Возврат (56%) и **коэффициент Шарпа** [25] из-за торговой стратегии, основанной на созданной модели, также много выше базовых показателей.

Джерри Чен, Аарон Чай, Мадхав Гоэль, Донован Лиу, Фаазила Мохамед, Дэвид Нам, Бонни Ву из Беркли проверяли гипотезу эффективного рынка [14] . На прогнозирование цен на акции на фондовом рынке из новостных статей влияет огромное количество источников. Внешние, а также внутренние факторы перемещают фондовый рынок. От чего-то малого как пресс-релиз конкретной фирмы, до нечта большего, как законы, принятые правительством, влияние на фондовый рынок иногда очень заметно. Гипотеза эффективного рынка (EMH) гласит, что цена на акции является отражением всех доступных источников информации, в том числе новостей, докладов, и поэтому невозможно использовать выгоду от переоценки активов или прогнозировать о будущих активах активов. Существуют сильные и слабые формы гипотезы, но, по сути, она подразумевает, что рынок всегда эффективен. Однако, критика теории предполагает, что это не всегда так. Исследователи проанализировали влияние разных информационных источников на стоимость акции и выяснили, что в целом влияние всегда

остается и сама теория эффективных рынков оставляет себе право на существование.

[15] Сяо-Цянь Сан, Хуа-Вэй Шэнь Сюэ-Ци Чэн, Юйцин Чжан. Китайская академия наук, Пекин, Китай. Существующие методы прогнозирования используют либо автокорреляцию цены акций, либо ее корреляцию с предложением и спросом на акции, или исследовать предвестники, экзогенные к рынку акций. В данной статье, используя транзакционную запись акций с идентификатором трейдеров, вводится новый индекс, чтобы охарактеризовать доверие к рынку, т. е. отношение количества трейдеров, которые активны в течение двух последовательных торговых дней к числу активных трейдеров в определенной торговле день. Сильная **причинность по Грейнджеру** [26] находится между индексом уверенности рынка и цены акции. Ученые также предсказывают цену акций путем включения индекса уверенности рынка в нейронную сеть на основе временных рядов цены акций. Экспериментальные результаты по 50 акций в двух китайских фондовых биржах демонстрируют, что точность прогнозирования цен акций улучшилась благодаря включению индекса уверенности в рынке. Это исследование проливает свет на применение междудневное торговое поведение для характеристики уверенности рынка и прогнозирования цены акций.

[16] Марк Данн. Колледж Корка. Прогнозирование фондового рынка. В этом отчете анализируется существующие и новые методы пронозирования. Применяются три подхода к проблеме: фундаментальный анализ, технический анализ и применение машинного обучения. В ходе работы находятся доказательства в поддержку слабой

формы гипотезы эффективного рынка, что теорическая цена не содержит полезной информации, но из выборочных данных можно получить предсказание. Фундаментальный анализ и машинное обучение могут использоваться для принятия решений инвесторами. Демонстрируется общий недостаток технического анализа и показывается, что он создает ограниченную полезную информацию.

[17] Амин Хедияти Мохаддама, Моин Хедайати Мохаддамб, Мортеза Эсфандьяри. Университет Тегерана. Прогнозирование индекса фондового рынка с использованием искусственной нейронной сети. В этом исследовании исллеедуется способность искусственной нейронной сети (ANN) в прогнозировании ежедневных цен фондовой биржи NASDAQ. Методология, используемая в этом исследовании, рассматривала краткосрочные исторические цены на акции а также день недели в качестве исходных данных. Ежедневные биржевые курсы NASDAQ с 28 января 2015 года по 18 июня 2015 года используются для разработки надежной модели. Первые 70 дней (с 28 января по 7 марта) выбираются как учебный набор данных и последние 29 дней используются для тестирования способности предсказания модели.

[18] Phuong Dang Toan Nguyen. Национальный Университет Сингапура. Прогнозирование волатильности. Волатильность жизненно важна на современном финансовом рынке. Это ключевой параметр для управления рисками, выбора портфеля, ценообразования производного от собственного капитала инструментов и особенно волатильности. Волатильность фондового рынка также имеет существенное значение для разработчиков политики, таких как центральные банки и финансовые регуляторы при создании правил и положений, денежно-

кредитной и налоговой политика. Предыдущие работы по прогнозированию волатильности пытались включить поток информации в существующие модели волатильности, либо от количества статей, объема или из новостных статей в качестве основного источника для прогнозирования волатильности. Более того, в предыдущих подходах используется только один метод машинного обучения, а сложный характер естественного языка является сложным для любого отдельного алгоритма. Чтобы устранить эти ограничения, автор предлагает гибридный подход, который сочетает в себе как популярную модель волатильности **GARCH** [27], так и информационный поток, извлеченный непосредственно из новостей. Следовательно, используется непосредственно два источника данных: исторические данные о ценах и данные новостной статьи. Также применяются ансамблевые методы для дальнейшего повышения производительности системы. Представляется подробное описание системы, которая реализует новый гибридную модель начиная от сбора данных, построения модели GARCH и заканчивая извлечением информационного потока из статей и применения методов машинного обучения для прогнозирования. В ходе различных экспериментов показывается, что гибридная модель по точности, далеко за пределами популярной модели семейства GARCH. Также показывается, что использование ансамблевых методов в гибридной модели приводит к большей точности, чем SVM. Результаты дают конкретное свидетельство относительно EMH, основы для многих современных финансовых моделей. В частности, подтверждается гипотезу о том, что поток информации требует времени для поглощения в процесс ценообразования. Прямое извлечение новой информации из новостных

статей в значительной степени улучшает эффективность модели прогнозирования волатильности.

ГЛАВА 1. Анализ методов глубинного обучения, библиотек и инструментов

Многие смотрели или слышали об играх между AlphaGo и профессиональным игроком Go Lee Sedol в 2016 году. Ли имеет самый высокий ранг (девятый дан) и многие победы на чемпионатах мира. Несомненно, он один из лучших игроков в го в мире, но он проиграл со счётом 1-4 против AlphaGo. До этого го считалась неразрешимой игрой для усвоения компьютерами, поскольку ее простые правила выставляют экспоненциальное количество вариаций на игровом поле, гораздо больше, чем в шахматах. Это событие наверняка выделило 2016 год как большой год для искусственного интеллекта (ИИ). Из-за AlphaGo большое внимание было уделено прогрессу ИИ.

Между тем, многие компании тратят ресурсы на то, чтобы расширить границы ИИ в разных приложениях, которые действительно могут изменить или даже революционизировать то, как мы собираемся жить. К знакомым примерам относятся самоуправляемые автомобили, чатботы, домашние помощники и многие другие. Одним из секретных рецептов прогресса, который мы имели в последние годы, является глубинное обучение.

Модели глубинного обучения, простыми словами, представляют собой большие и глубокие искусственные нейронные сети. Нейронная сеть (НС) может быть хорошо представлена в виде ориентированного

ациклического графа: входной слой принимает сигнальные векторы; один или несколько скрытых слоев обрабатывают выходы предыдущего слоя. Первоначальная концепция нейронной сети может быть прослежена более полувека назад. Возникает резонный вопрос: «Почему именно сейчас?».

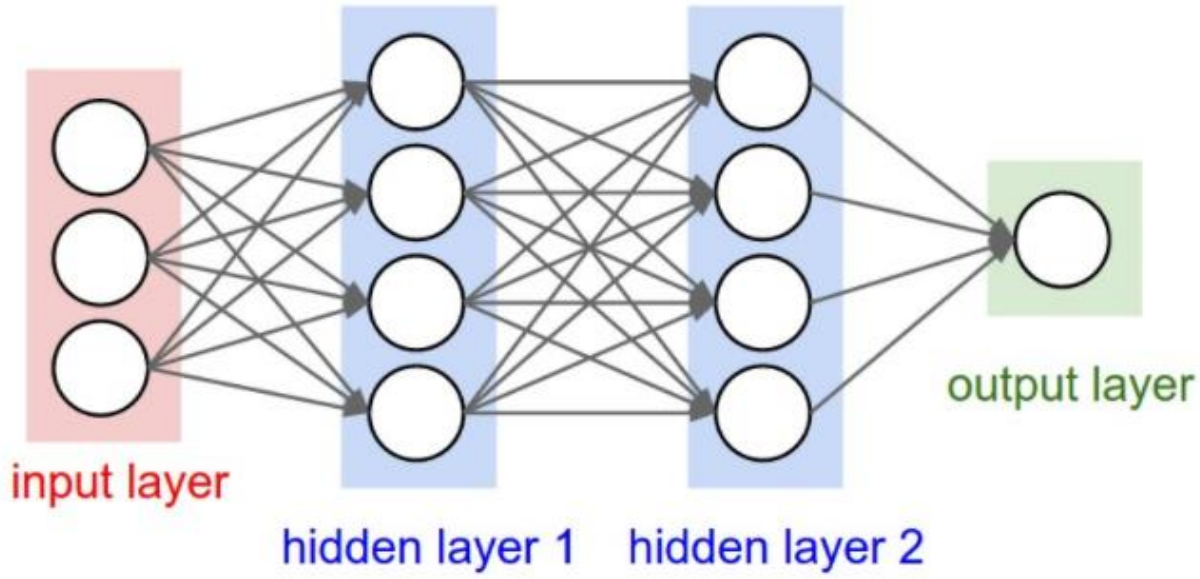


Рис. 1. Трехслойная искусственная нейронная сеть.

Причина довольно проста:

- Экспоненциальный рост объема данных благодаря распространению доступа к высокоскоростному интернету;
- Удешевление компонентов вычислительной техники при росте их производительности.

Большая и глубокая нейронная сеть имеет гораздо больше слоев + еще много узлов в каждом слое, что приводит к экспоненциальному росту множества параметров для настройки. Без достаточного количества данных мы не сможем эффективно изучить параметры. Без мощных

компьютеров обучение было бы слишком медленным и недостаточным. Появился целый класс микропроцессоров и сопроцессоров, которых называют **нейронными** или **ИИ-ускорителями (AI accelerator)** [55], который используется для аппаратного ускорения работы алгоритмов машинного обучения и нейронных сетей благодаря высочайшему уровню параллелизма:

- **Тензорные процессоры** – как правило сопроцессора, оперирующие тензорами: Google TPU, Intel Nervana NNP.
- **Нейроморфные процессоры** – многоядерные процессора с тысячами вычислительных элементов, каждый из которых в свою очередь эмулирует тысячи нейронов: IBM TrueNorth, Neural Engine, Eyeriss, SpiNNaker, Zeroth NPU.
- **Процессоры машинного зрения** – схожи с тензорными процессорами, но ставка деляется на **CV (Computer Vision)**: Intel Movidius Myriad 2, Mobileye EyeQ.

Все они позволяют проводить обучение практически в реальном времени, но пока сохраняется тенденция к применению графических процессоров (GPU). Последние продукты**: Volta GV100** (GPU от Nvidia, содержащий специальные ядра для тензорных вычислений), **Nvidia DGX-1** (сервер построенный на GV100), **Radeon Instinct** (продукт от AMD, аналог GV100) [55].

При небольшом наборе данных традиционные алгоритмы (регрессия, случайные леса, SVM, GBM и т. д.) или статистическое обучение отлично справляются, но как только масштаб данных приближается к террабайтам и петабайтам, большая НС демонстрирует своё превосходство. Отчасти потому, что по сравнению с традиционной

моделью ML модель нейронной сети имеет гораздо больше параметров и имеет возможность изучать сложные нелинейные модели. Таким образом, мы ожидаем, что модель сама выберет наиболее полезные связи, не привлекая слишком много специализированных экспертных знаний, сделанных человеком вручную.

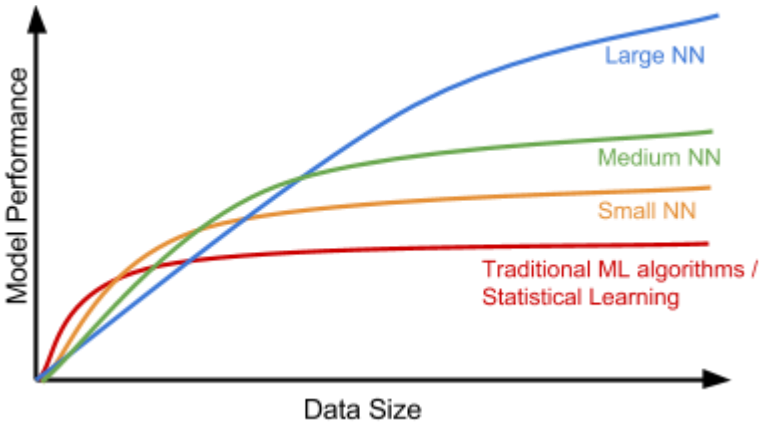


Рис. 2. Зависимость производительности модели от роста объёма данных.

1.1. Свёрточная нейронная сеть (CNN)

Свёрточные нейронные сети (Convolutional Neural Network, CNN), представляют собой тип искусственных нейронных сетей с обратной связью, в которых схема взаимодействия между нейронами вдохновлена организацией системы визуальной коры. Первичная зрительная кора (V1) обнаруживает края объекта из исходного визуального ввода из сетчатки. Вторичная визуальная кора (V2) получает краевые признаки от V1 и извлекает простые визуальные свойства, такие как ориентация, пространственная частота и цвет. Визуальная область V4 (нижняя височная извилина) обрабатывает более сложные атрибуты объекта. Все обработанные визуальные признаки поступают в конечную

в область для распознавания объектов. Скращенный путь между V1 и V4 побудил к созданию особого типа CNN с соединениями между несмежными слоями: **Residual Net**, содержащей «Остаточный блок», который хранит некоторый ввод одного слоя, который должен быть передан на два слоя дальше.

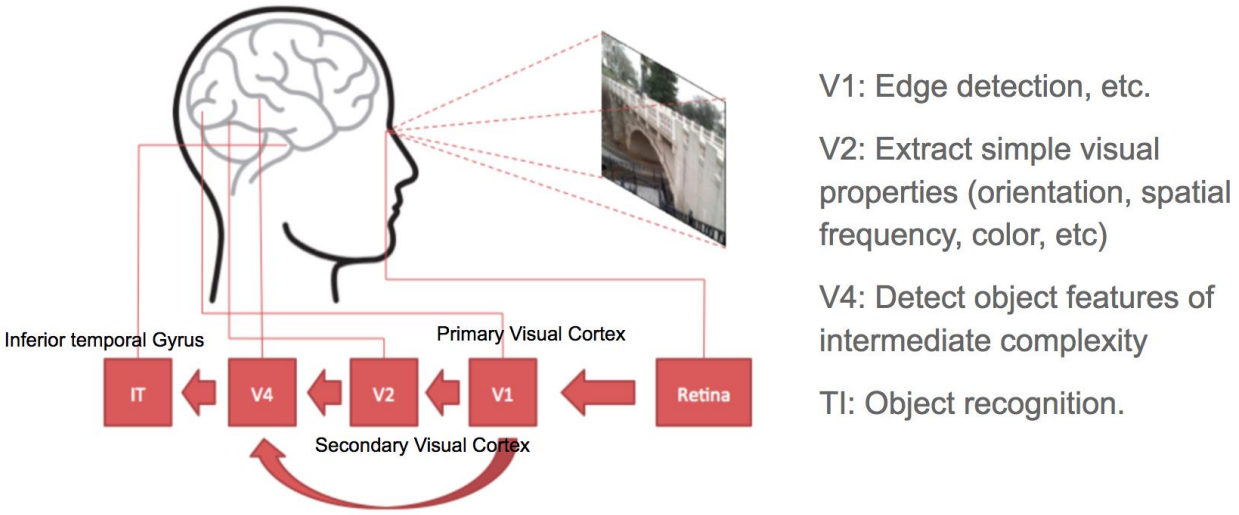


Рис. 3.Иллюстрация визуальной коры человеческого мозга [29].

Свертка является математическим термином, здесь речь идет об операции между двумя матрицами. Сверточный слой имеет определенную небольшую матрицу, также называемую ядром или фильтром. По мере того, как ядро скользит или свертывается по матричному представлению входного изображения, оно вычисляет поэтапное умножение значений в матрице ядра и исходных значений изображения. Специально разработанные ядра могут обрабатывать изображения для общих целей, таких как размытие, резкость, обнаружение краев и многие другие, быстро и эффективно.

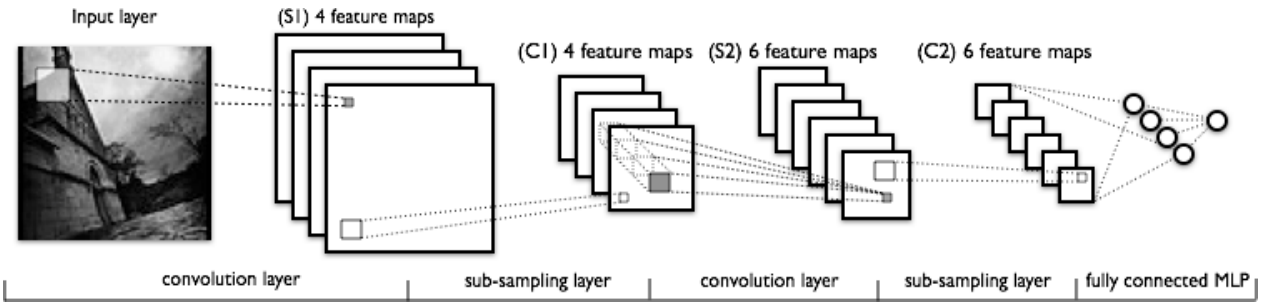


Рис. 4. Архитектура LeNet состоит из двух наборов сверточных, активационных и объединяющих слоев, за которыми следует полносвязный уровень, активация, еще один полносвязный уровень и, наконец, softmax-классификатор.

Сверточный и объединяющий (pooling или sub-sampling на рис. 4) слои действуют как V1, V2 и V4 системы, реагируя на извлечение признаков. Распознавание признаков объектов происходит в более поздних полносвязных слоях, которые потребляют извлеченные признаки.

1.2. Рекуррентная нейронная сеть (RNN)

Последовательная модель обычно предназначена для преобразования входной последовательности одной предметной области в выходную последовательность другой. **Рекуррентная нейронная сеть (Recurrent Neural Network, RNN),** подходит для этой цели и продемонстрировала прорыв в решении таких задач, как распознавание рукописного текста, распознавание речи и машинный перевод.

RNN создавалась с возможностью обработки длинных последовательных данных и решения задач с распределением контекста во времени. Модель обрабатывает один элемент в последовательности на одном временном шаге. После вычисления обновленное состояние передается на следующий шаг во времени, чтобы облегчить вычисление следующего элемента. Представьте себе случай, когда RNN читает все

статьи в Википедии посимвольно, а затем может предсказать следующие слова с учетом контекста.

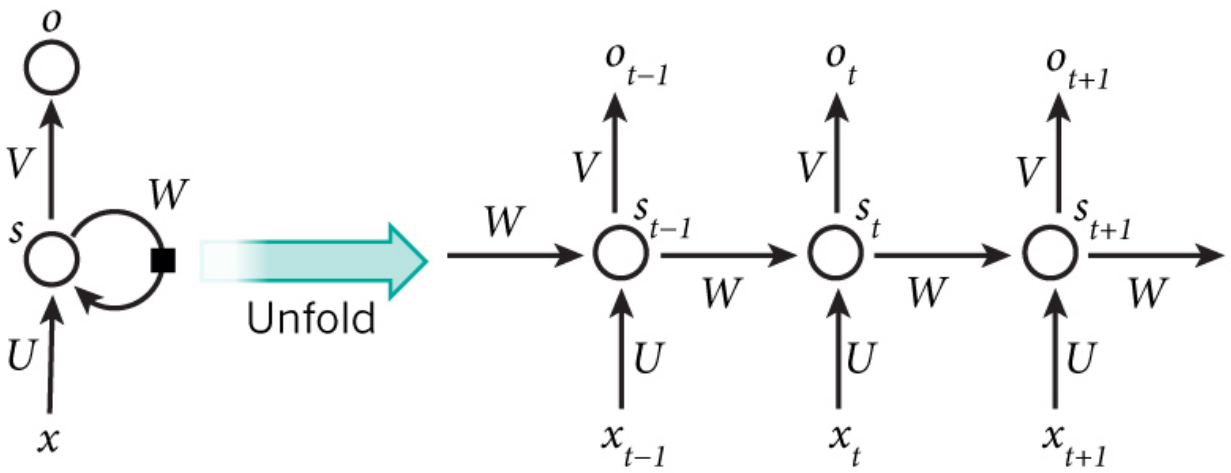


Рис. 5. Рекуррентная нейронная сеть с одним скрытым элементом (слева) и ее разворачивающейся версией во времени (справа). Развернутая версия иллюстрирует, что происходит во времени: s_{t-1}, s_t и s_{t+1} - это одна и та же единица с разными состояниями на разных временных шагах ***t - 1***, ***t*** и ***t + 1*** [30].

Однако простые персептронные сети, которые линейно объединяют текущий элемент ввода и последний элемент вывода, могут легко потерять долгосрочные зависимости. Например, мы начинаем предложение с «Алисой работает в...» и позже после целого абзаца мы хотим начать предложение с «Она» или «Он» правильно определив пол. Если модель забывает имя персонажа, то мы никогда не сможем этого добиться. Чтобы решить эту проблему, исследователи создали специальный нейрон с гораздо более сложной внутренней структурой для запоминания долгосрочного контекста, называемого ячейкой **Long-short Term Memory (LSTM)**. Он достаточно умен, чтобы узнать, как долго он должен запоминать старую информацию, когда использовать новые данные и как объединить старую память с новым входом [31].

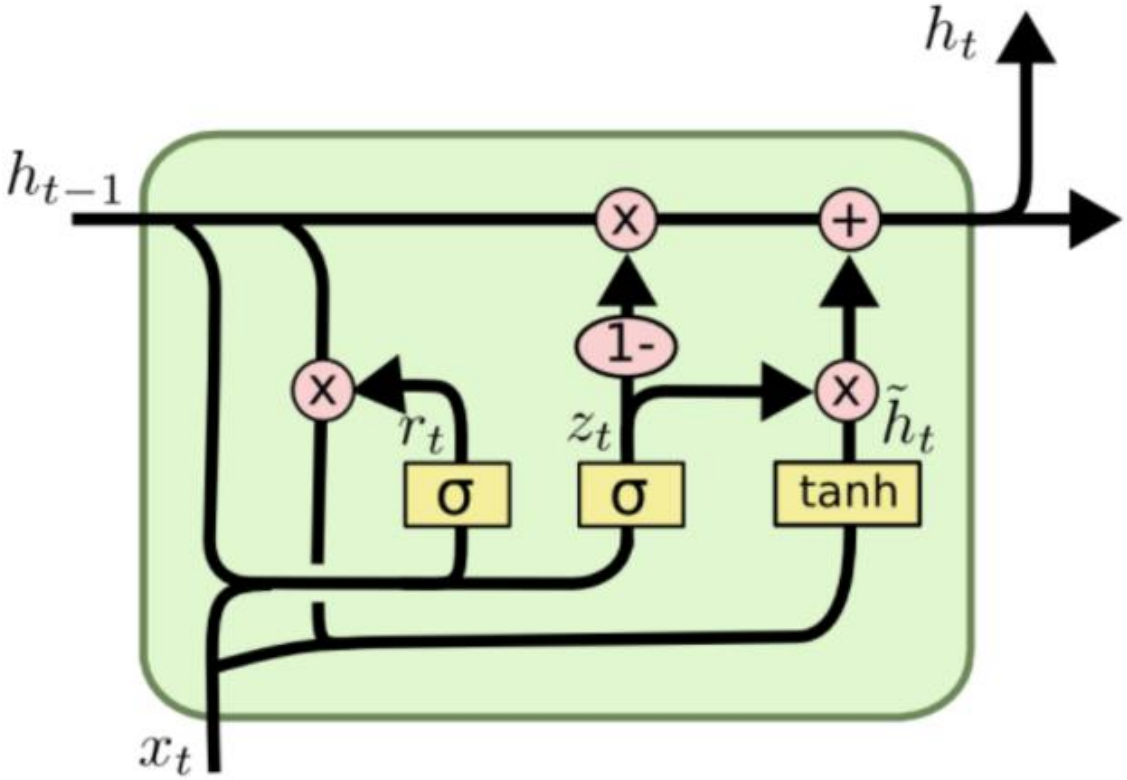


Рис. 6. Структура LSTM нейрона [31].

Чтобы продемонстрировать мощь RNN, Андрей Карпаты [32] построил символьную языковую модель, используя RNN с ячейками LSTM. Не зная английского словарного запаса заранее, модель могла изучить взаимосвязь между символами, чтобы сформировать слова, а затем связь между словами, чтобы сформировать предложения. Она смогла достичь достойной производительности даже без огромного набора данных для обучения.

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.
```

Рис. 7. Символьная RNN пишет текст в духе Шекспира [32].

1.3. RNN: Sequence-to-Sequence

Модель **Sequence-to-Sequence** [33] представляет собой расширенную версию RNN, но поле её приложений достаточно хорошо различимо. Так же, как и RNN, модель Sequence-to-Sequence работает с последовательными данными, но чаще она используется для разработки чатботов или персональных помощников, чтобы создать осмысленный ответ для входного вопроса. Модель Sequence-to-Sequence состоит из двух RNN, кодировщика и декодера. Кодировщик изучает контекстуальную информацию из входных слов, а затем передает знания на сторону декодера через «вектор контекста» (или «вектор мышления», как

показано на рис. 8). Наконец, декодер потребляет вектор контекста и генерирует правильные ответы.

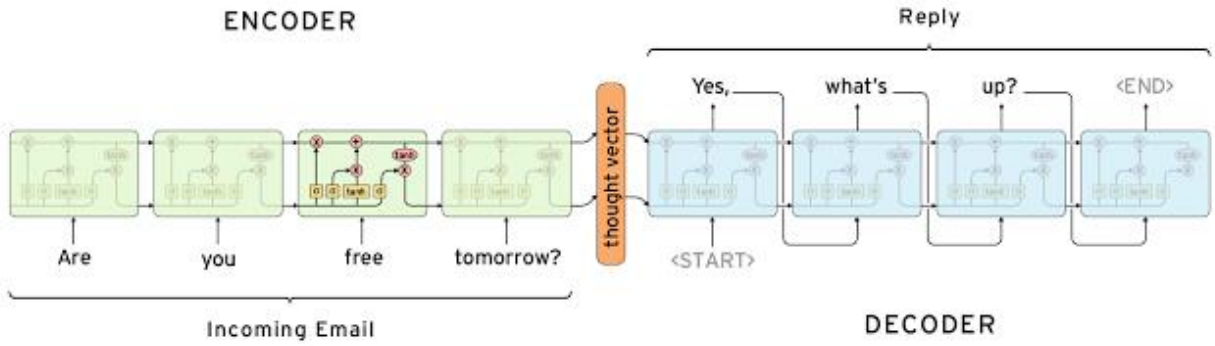


Рис. 8. Sequence-to-Sequence для генерации автоматических ответов Gmail [33].

1.4. Автокодировщики

В отличие от предыдущих моделей, **автокодировщики** предназначены для обучения без учителя. Они предназначен для изучения **низкоразмерного** представления **высокоразмерного** набора данных, аналогично тому, что делает **метод главных компонентов (Principal component analysis, PCA)** [35]. Модель автокодировщика пытается обучить функцию аппроксимации $f(x) \approx x$ для воспроизведения входных данных. Однако она ограничен узким слоем в середине с очень небольшим количеством узлов. При ограниченной емкости модель вынуждена формировать очень эффективное кодирование данных.

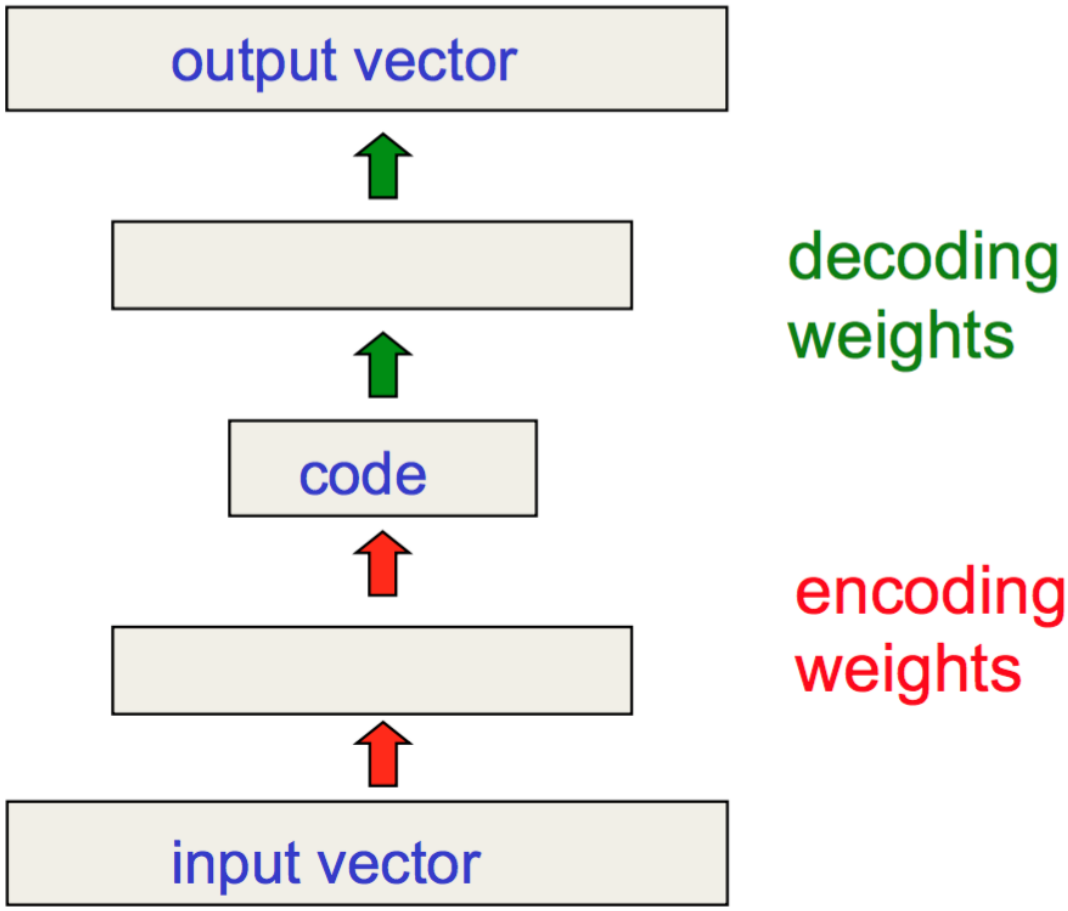


Рис. 9. Модель автокодера имеет узкий слой с несколькими нейронами.

Хинтон и Салахутдинов [36] использовали автокодировщики для сжатия документов по различным темам. Как показано на рис. 10, когда и PCA, и автокодировщик были применены для уменьшения документов на два измерения, автокодировщик показал гораздо лучший результат. С помощью этой модели мы можем эффективно выполнять сжатие данных, чтобы ускорить поиск информации, включая документы и изображения.

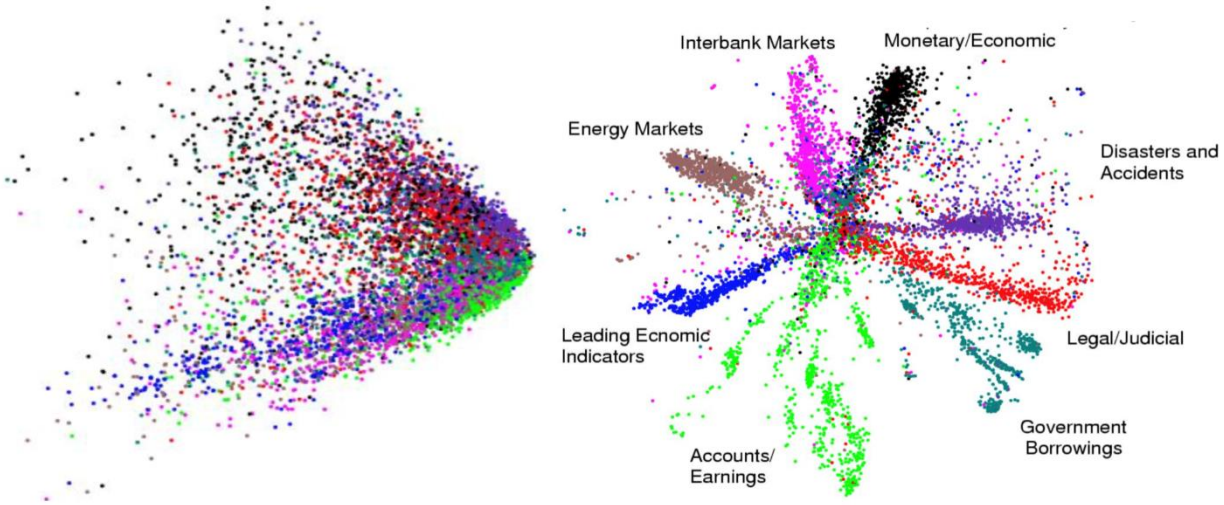
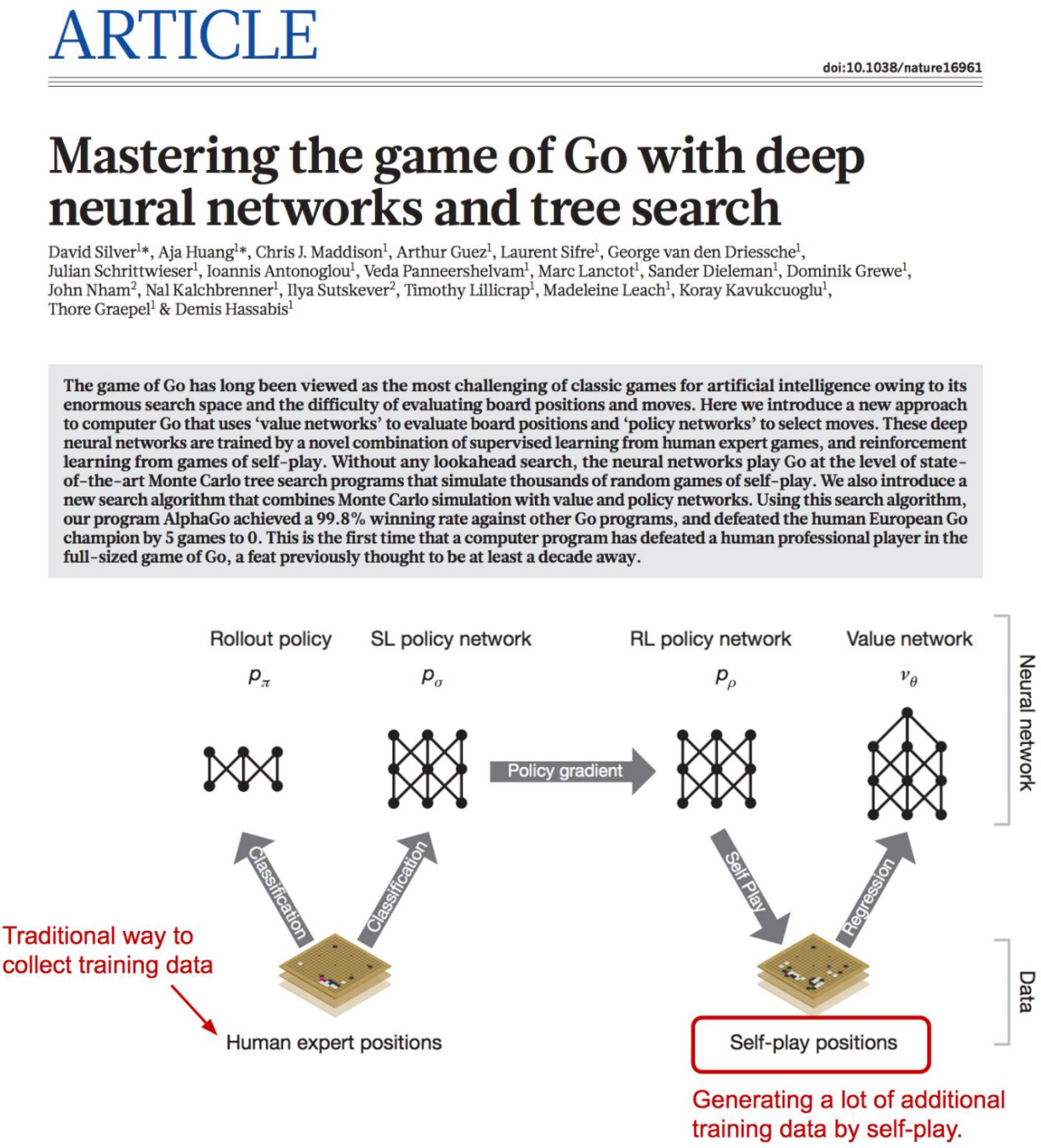


Рис. 10. Результаты работы PCA (слева) и автокодировщика (справа), когда оба пытаются сжать документы в двумерное представление [36].

1.5. Обучение с подкреплением

Поскольку глава началась с упоминания AlphaGo, то стоит рассмотреть причину его победы над человеком. **Обучение с подкреплением (Reinforcement Learning, RL)** [37] является одним из секретов его успеха. RL - это подобласть машинного обучения, которая позволяет машинам и программным агентам автоматически определять оптимальное поведение в заданном контексте с целью максимизировать долгосрочную производительность, измеряемую указанным показателем.



ARTICLE

doi:10.1038/nature16961

Mastering the game of Go with deep neural networks and tree search

David Silver[1]*, Aja Huang[1]*, Chris J. Maddison[1], Arthur Guez[1], Laurent Sifre[1], George van den Driessche[1], Julian Schrittwieser[1], Ioannis Antonoglou[1], Veda Panneershelvam[1], Marc Lanctot[1], Sander Dieleman[1], Dominik Grewe[1], John Nham[2], Nal Kalchbrenner[1], Ilya Sutskever[2], Timothy Lillicrap[1], Madeleine Leach[1], Koray Kavukcuoglu[1], Thore Graepel[1] & Demis Hassabis[1]

The game of Go has long been viewed as the most challenging of classic games for artificial intelligence owing to its enormous search space and the difficulty of evaluating board positions and moves. Here we introduce a new approach to computer Go that uses 'value networks' to evaluate board positions and 'policy networks' to select moves. These deep neural networks are trained by a novel combination of supervised learning from human expert games, and reinforcement learning from games of self-play. Without any lookahead search, the neural networks play Go at the level of state-of-the-art Monte Carlo tree search programs that simulate thousands of random games of self-play. We also introduce a new search algorithm that combines Monte Carlo simulation with value and policy networks. Using this search algorithm, our program AlphaGo achieved a 99.8% winning rate against other Go programs, and defeated the human European Go champion by 5 games to 0. This is the first time that a computer program has defeated a human professional player in the full-sized game of Go, a feat previously thought to be at least a decade away.

Рис. 11. Пайплайн и архитектура обучения нейронной сети AlphaGo [38].

Система AlphaGo начинается с обучения с учителем, чтобы обучиться стратегии быстрого развертывания и сети правил, опираясь на подготовленный вручную учебный набор игр, сыгранных

профессионалами. Она узнает лучшую стратегию в текущей игровой позиции. Затем она применяет обучение с подкреплением, играя сама с собой. Сеть правил RL улучшается, когда она выигрывает все больше и больше игр против предыдущих версий сети. На этапе самостоятельной игры AlphaGo становится все сильнее и сильнее, играя против себя, не требуя дополнительных данных извне для обучения.

1.6. Генеративно-состязательная сеть

Генеративно-состязательная сеть (Generative Adversarial Network, GAN) [39], представляет собой один из типов глубоких генеративных моделей. GAN может создавать новые примеры после изучения реальных данных. Она состоит из двух моделей, конкурирующих друг с другом в игровой системе с нулевой суммой.

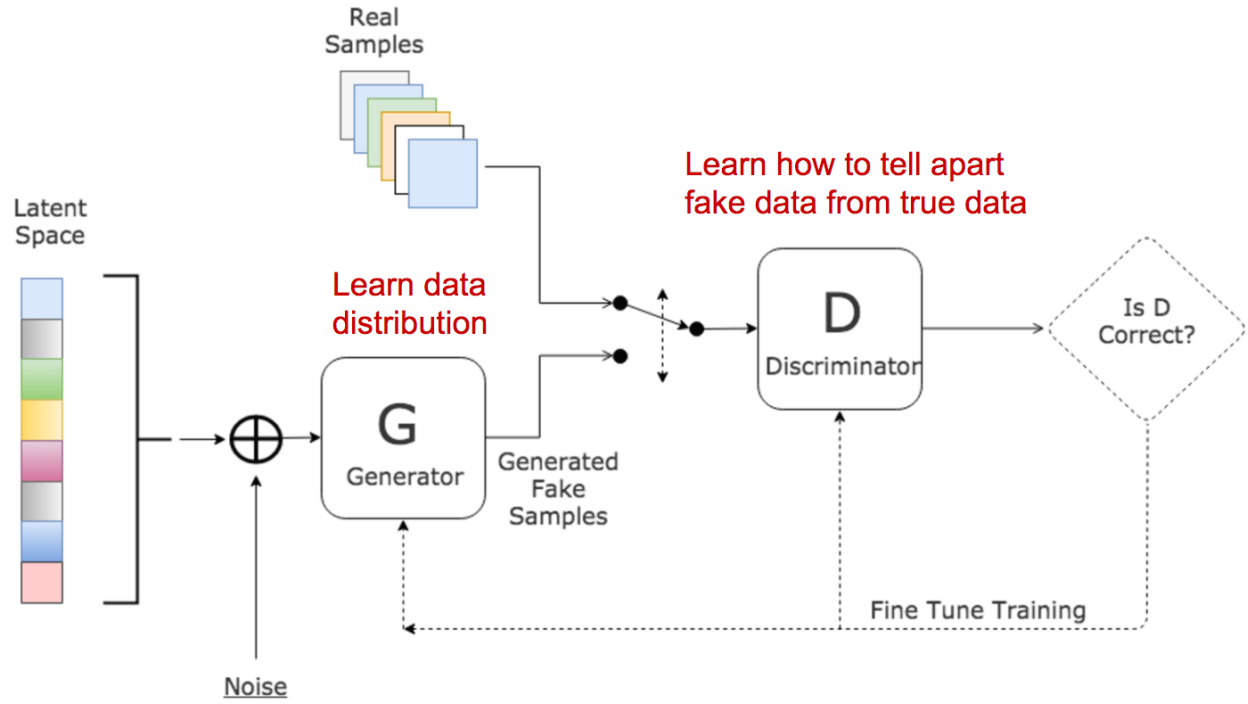


Рис. 12. Архитектура генеративно-состязательной сети [40].

В оригинальной статье GAN [39] было предложено генерировать осмысленные изображения после изучения реальных фотографий. GAN включает две независимые модели: **генератор** и **дискриминатор**. Генератор создает поддельные изображения и отправляет результат в модель дискриминатора. Дискриминатор работает как судья, поскольку он оптимизирован для определения реальных фотографий от поддельных. Модель генератора изо всех сил пытается обмануть дискриминатора, в то время как дискриминатор изо всех сил пытается не быть обманутым. Игра с нулевой суммой между двумя моделями мотивирует их развивать свои навыки и улучшать функциональность. После обучения мы берем модель генератора для создания новых изображений.

1.7. Библиотеки и инструменты

Изучив все эти модели, можно начать задаваться вопросом как реализовать и использовать их для реальных задач. Уже сейчас существует множество инструментов и библиотек с открытым исходным кодом для создания моделей глубокого обучения. Краткое сравнение самых популярных фреймворков [56][57][58] приведено в табл. 1:

Фреймворк	Распределенное выполнение	Архитектурные оптимизации	Визуализация	Поддержка сообществом	Портативность
Tensorflow	XX	XX	XX	XX	XX
PyTorch	XX	XX	XX	XX	XX
CNTK	XX	XX	X	-	XX
MXNet	X	XX	X	-	XX
Torch	-	XX	X	X	x
Caffe2	XX	XX	-	-	XX
Caffe	-	XX	X	X	X

Theano	-	XX	X	X	X

Табл. 1. Резюме по самым популярным фреймворкам для машинного обучения.

Большинство представленных фреймворков представляют модель в виде **вычислительного графа**, статического или динамического:

- Фреймворки, объявляющие графы вычислений статически, такие как TensorFlow, позволяют вам сначала определить архитектуру графа, а затем вы можете выполнить его много раз, прогнав через него некоторый объем данных. Это позволяет легко распределять задачи для выполнения на разных машинах. В дополнение к этому, фреймворки могут оптимизировать граф для вас заранее. Они также позволяют пользователям сериализовать график после его создания и выполнять его, не требуя кода, который его построил.
- Фреймворки, объявляющие графы вычислений динамически, такие как PyTorch, позволяют вам неявно определять вычислительный граф. Структуры динамического графа обычно менее инвазивные и глубоко интегрируются с используемым языком программирования. Это приводит к более чистыму и легко отлаживаемому коду и позволяет использовать условные операторы и циклы для построения более сложных графовых структур, таких как рекуррентные нейронные сети. Построение и исполнение графа переплетаются, что оставляет мало времени для его оптимизации.

В конечном итоге для обучения была выбрана архитектура рекуррентной нейронной сети (RNN), построенной на элементах долгой

краткосрочной памяти (LSTM). Поскольку история движения цен представляет собой временной ряд, то нам необходимо регулярно обращаться к ней и учитывать долгосрочный контекст для чего, собственно, RNN с LSTM и предназначены.

Для реализации выбранной модели выбор пал на TensorFlow. Популярность данного фреймворка вполне заслужена: мы получаем поддержку большого числа разработчиков, обширный выбор документации, открытый исходный код, распределенное выполнение наших моделей, различные архитектурные оптимизации, визуализацию во время выполнения (что упрощает отладку модели) и портативность вплоть до того, что сможем запускать модель на смартфонах с Android.

ГЛАВА 2. Подготовка данных и построение предсказательной модели

Далее мы разберем как построить модель RNN с ячейками LSTM для прогнозирования индекса S&P500.

2.1. Подготовка данных

Набор данных можно загрузить с Yahoo! Finance ^GSPC [43]. В данной работе использовались данные S&P 500 с **3 января 1950 года** (максимальная дата, которую Yahoo! Finance может отследить) до **23 июня 2017 года**. Набор данных предоставляет несколько ценовых точек в день. Для простоты мы будем использовать только ежедневные цены закрытия для прогнозирования. **TensorBoard** [44] будет использоваться для отладки и отслеживания модели.

Напоминание: **рекуррентная нейронная сеть (RNN)** представляет собой тип искусственной нейронной сети с рекуррентными переходами в скрытых слоях, что позволяет ей использовать предыдущее состояние скрытых нейронов для получения нового результата с учетом нового ввода. RNN умеет обрабатывать последовательные данные. **Долгая краткосрочная память (LSTM)** - специально разработанная ячейка, которая помогает RNN лучше запоминать долгосрочный контекст.

Цены на акции представляют собой временные ряды длины N, определенные как $p_0, p_1, ..., p_{N-1}$, в которых p_i - цена закрытия в день i, $0 \le i < N$. Представьте, что у нас есть скользящее окно фиксированного размера w (позже мы будем называть это как `input_size`), и мы перемещаем его вправо на w, чтобы не было перекрытия между данными во всех скользящих окнах.

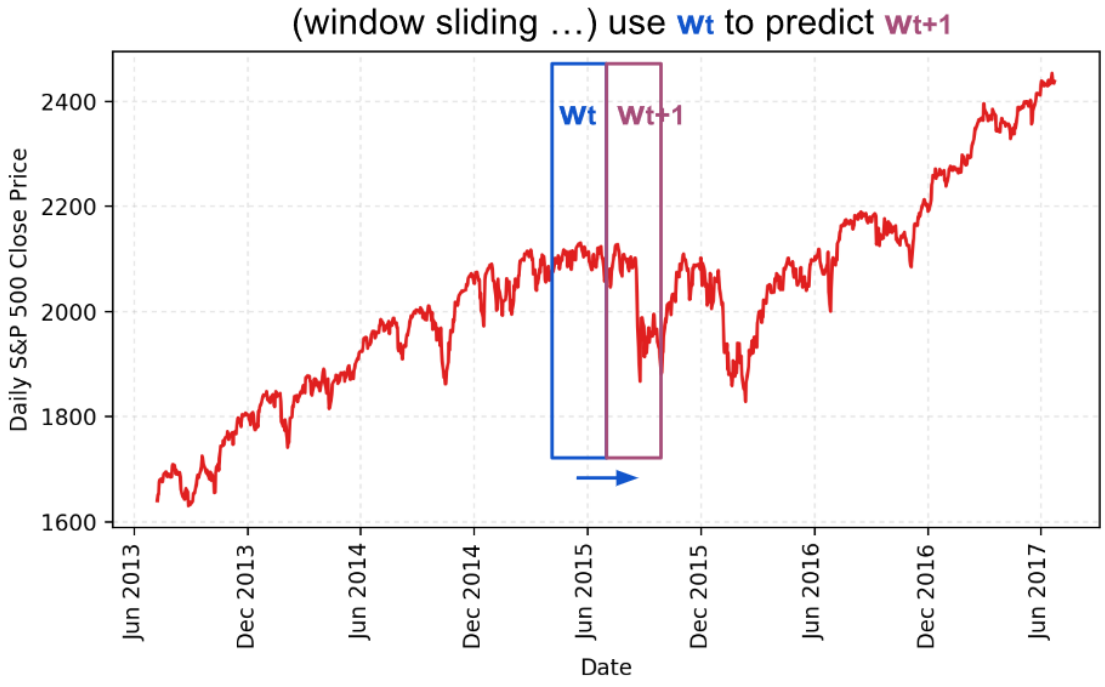


Рис. 13. Цены S&P 500 во времени. Мы используем данные в одном скользящем окне, чтобы сделать прогноз для следующего, в то время как между двумя последовательными окнами нет перекрытия.

Модель RNN, которую мы строим, имеет LSTM-ячейки в качестве основных скрытых элементов. Мы используем значения с самого начала обучения в первом скользящем окне W_0 до окна W_t в момент времени t:

$$W_0 = (p_0, p_1, \ldots, p_{w-1})$$
$$W_1 = (p_w, p_{w+1}, \ldots, p_{2w-1})$$
$$\ldots$$
$$W_t = (p_{tw}, p_{tw+1}, \ldots, p_{(t+1)w-1})$$

для прогнозирования цен в следующем окне W_{t+1}:

$$W_{t+1} = (p_{(t+1)w}, p_{(t+1)w+1}, \ldots, p_{(t+2)w-1})$$

По существу мы пытаемся найти функцию аппроксимации, $f(W_0, W_1, \ldots, W_t) \approx W_{t+1}$.

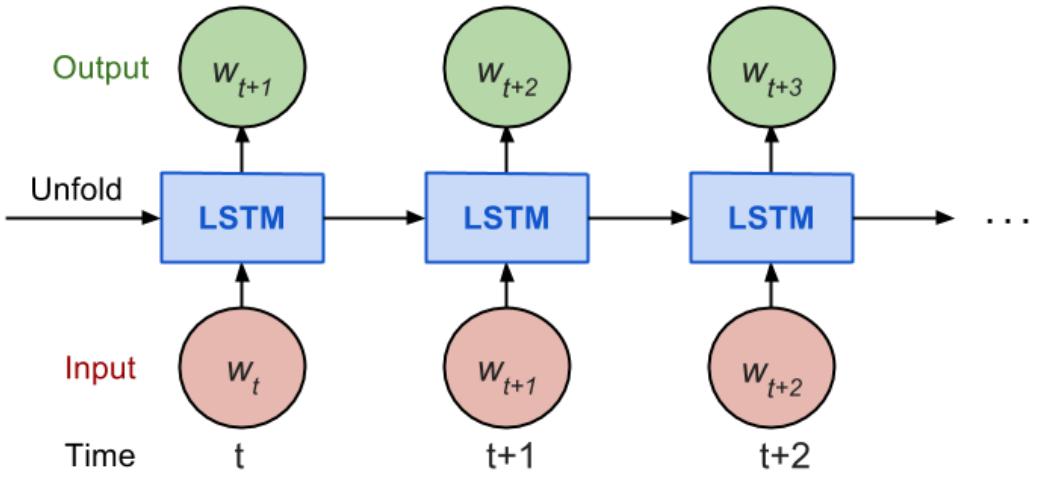


Рис. 14. Развернутая версия RNN.

Рассматривая, как работает **обратное распространение во времени (Backpropagation through time, BPTT)** [45], мы обычно обучаем RNN в «развернутой» версии, так что нам не нужно проводить обратное распространение слишком далеко по времени и увеличивать сложность обучения.

Вот объяснение по `num_steps` из документации Tensorflow:

Выход рекуррентной нейронной сети (RNN) зависит от произвольно отдаленных входов. К сожалению, это затрудняет вычисления обратного распространения. Для того, чтобы сделать процесс обучения приемлемым, обычной практикой является создание «развернутой» версии сети, которая содержит фиксированное число (`num_steps`) входов и выходов LSTM. Затем модель обучается в этом конечном приближении RNN. Это может быть реализовано путем подачи входных данных длиной `num_steps` за один раз и выполнения обратного прохода после каждого такого входного блока.

Последовательность цен сначала разделяется на небольшие неперекрывающиеся окна. Каждое из них содержит `input_size` числа, каждое из которых рассматривается как один независимый элемент ввода. Затем любые последовательные элементы ввода `num_steps` группируются в один ввод для обучения, образуя «**нераскрученную**» версию RNN для обучения в TensorFlow. Соответствующая метка является элементом ввода сразу после них.

Например, если `input_size = 3` и `num_steps = 2`, то первые несколько примеров обучения будут выглядеть так:

$$\begin{aligned}
\text{Input}_1 &= [[p_0, p_1, p_2], [p_3, p_4, p_5]], \text{Label}_1 = [p_6, p_7, p_8] \\
\text{Input}_2 &= [[p_3, p_4, p_5], [p_6, p_7, p_8]], \text{Label}_2 = [p_9, p_{10}, p_{11}] \\
\text{Input}_3 &= [[p_6, p_7, p_8], [p_9, p_{10}, p_{11}]], \text{Label}_3 = [p_{12}, p_{13}, p_{14}]
\end{aligned}$$

Поскольку мы всегда хотим предсказать будущие значения, мы берем последние **10%** данных в качестве тестовых.

2.2. Нормализация данных

Индекс S&P500 увеличивается со временем, что приводит к тому, что большинство значений в тестовом наборе выходит за допустимые пределы, и, таким образом, модель должна предсказать значения, которые она никогда не видела раньше. Печально и неудивительно, что рано или поздно модель перестает вести себя адекватно (рис. 15).

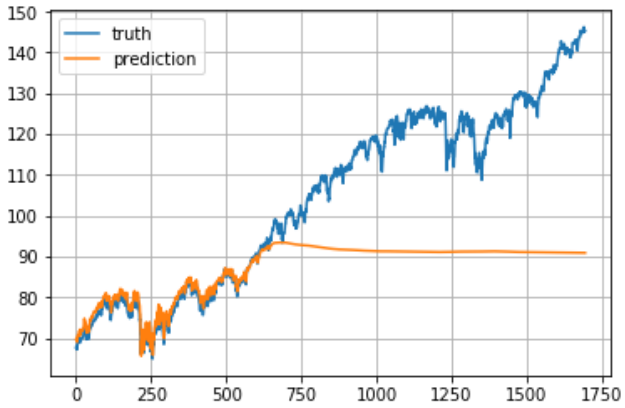


Рис. 15. Пример, когда RNN пытается предсказывать значения, лежащие за пределами обучающих данных.

Чтобы решить проблему с масштабом, необходимо нормализовать цены в каждом скользящем окне. Теперь задачей становится предсказать относительные уровни изменения вместо абсолютных значений. В нормализованном скользящем окне W_t в момент времени t все значения делятся на последнюю неизвестную цену - последнюю цену в W_{t-1}:

$$W'_t = \left(\frac{p_{tw}}{p_{tw-1}}, \frac{p_{tw+1}}{p_{tw-1}}, \ldots, \frac{p_{(t+1)w-1}}{p_{tw-1}}\right)$$

2.3. Построение модели

Определения, которые будут использоваться в модели (табл. 2):

lstm_size	Число элементов в одном LSTM-слое
num_layers	Число LSTM-слоёв
keep_prob	Процент элементов для сохранения в операции исключения [46]
init_learning_rate	Начальная скорость обучения
learning_rate_decay	Уровень затухания скорости обучения в конечных эпохах обучения [47]
init_epoch	Число эпох, использующих константу init_learning_rate
max_epoch	Общее число эпох обучения
input_size	Размер скользящего окна / одна точка обучающих данных
batch_size	Количество точек данных для использования в одном мини-батче (mini-batch)
embedding_size	Контролирует размер каждого вектора вложения [48]
stock_count	Числу уникальных акций в наборе данных

Табл. 2. Определения модели в TensorFlow.

LSTM-модель имеет num_layers комбинированных слоёв LSTM, и каждый слой содержит lstm_size LSTM-элементов. Затем к выходу

каждого LSTM-элемента применяется маска исключения [46] с вероятностью сохранения `keep_prob`. Цель исключения состоит в том, чтобы устранить потенциальную сильную зависимость от одного измерения, чтобы предотвратить переобучение.

Обучение требует `max_epoch` эпох в целом; **эпоха** [47]- это один полный проход по всем обучающим данным. За одну эпоху обучающие данные разбиваются на мини-батчи размера `batch_size`. Мы отправляем один мини-батч в модель для одного обучения по BPTT. Скорость обучения устанавливается на `init_learning_rate` в течение первых эпох `init_epoch`, а затем уменьшается на × `learn_rate_decay` в каждую последующую эпоху.

Ожидается, что модель изучит ценовые последовательности во времени разных компаний. Из-за различных шаблонов хотелось бы, чтобы модель явно говорила с какими акциями она работает. **Вложение** [48] выгодно в сравнии с **one-hot encoding**, потому что:

- Учитывая, что в обучающие данные входит N акций, one-hot encoding приведет к дополнительным разреженным N (или $N-1$) параметрам. Как только каждая акция отображается на гораздо меньший вектор вложения длины k, $k \ll N$, мы получаем гораздо более сжатое представление и меньший набор данных.
- Поскольку векторы вложения являются переменными для обучения, то схожие акции могут быть связаны с аналогичными вложениями и помогают прогнозировать друг друга, такие как «GOOG» и «GOOGL».

В рекуррентной нейронной сети на одном временном шаге ***t***, входной вектор содержит `input_size` (помеченные как ***W***) суточные значения цены ***i***-й акции, $(p_i, t_w, p_i, t_{w+1}, ..., p_{i,(t+1)w-1})$. Акция однозначно отображается на вектор длины `embedding_size` (помечен как ***k***), $(e_i, 0, e_i, 1, ..., e_i, k)$, Как показано на рис. 16, вектор цены конкатенируется с вектором вложения, а затем подается в ячейку LSTM.

Другой альтернативой является конкатенация векторов вложения с последним состоянием ячейки LSTM и получением новых весов ***W*** и смещение ***b*** в выходном слое. Однако, таким образом, LSTM-элемент не может отличить цены одной акции от другой, и его мощность будет в значительной степени сдерживаться.

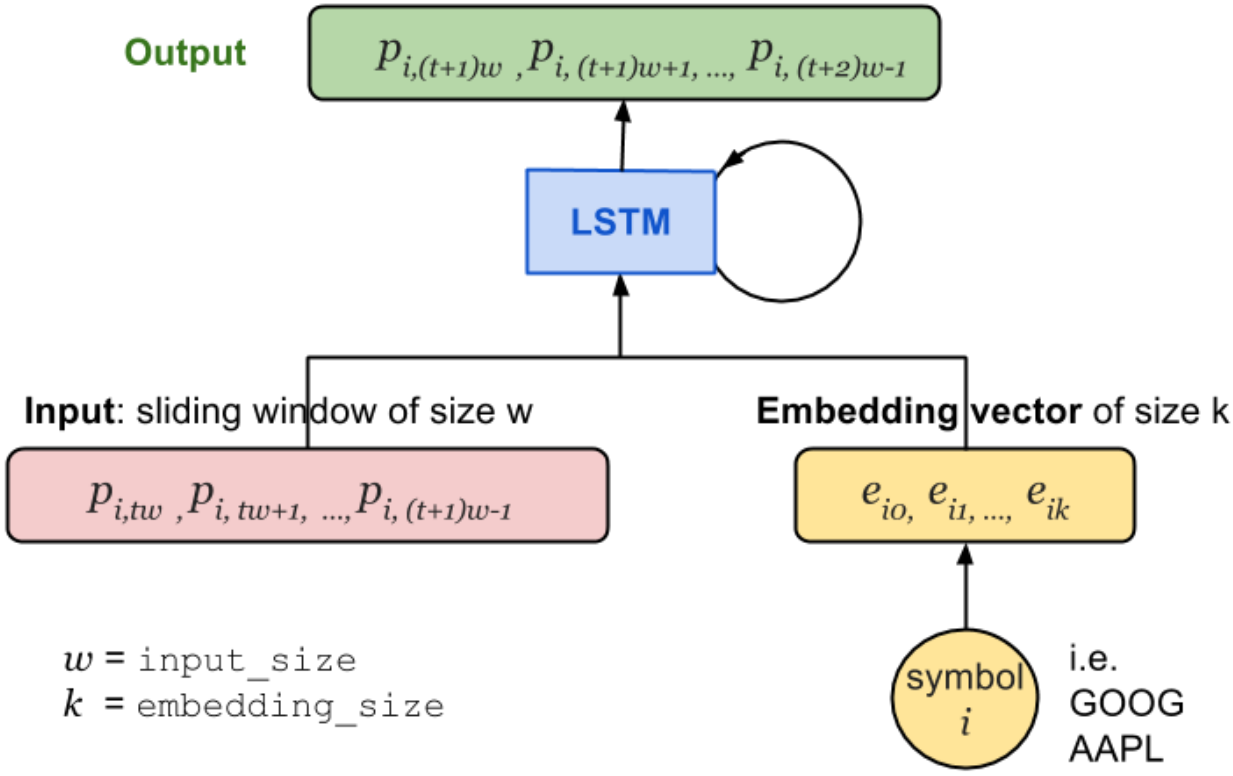


Рис. 16. Архитектура модели RNN для прогнозирования цен на акции с вложением.

ГЛАВА 3. Обучение, визуализация и результаты

3.1. Эксперименты

В первом эксперименте использована следующая конфигурация (табл. 3):

num_layers	1
keep_prob	0.8
batch_size	64
init_learning_rate	0.001
learning_rate_decay	0.99
init_epoch	5
max_epoch	100
num_steps	30

Табл. 3. Эксперимент #1.

В целом прогнозирование цен на акции - непростая задача. Особенно после нормализации ценовые тенденции выглядят очень шумно.

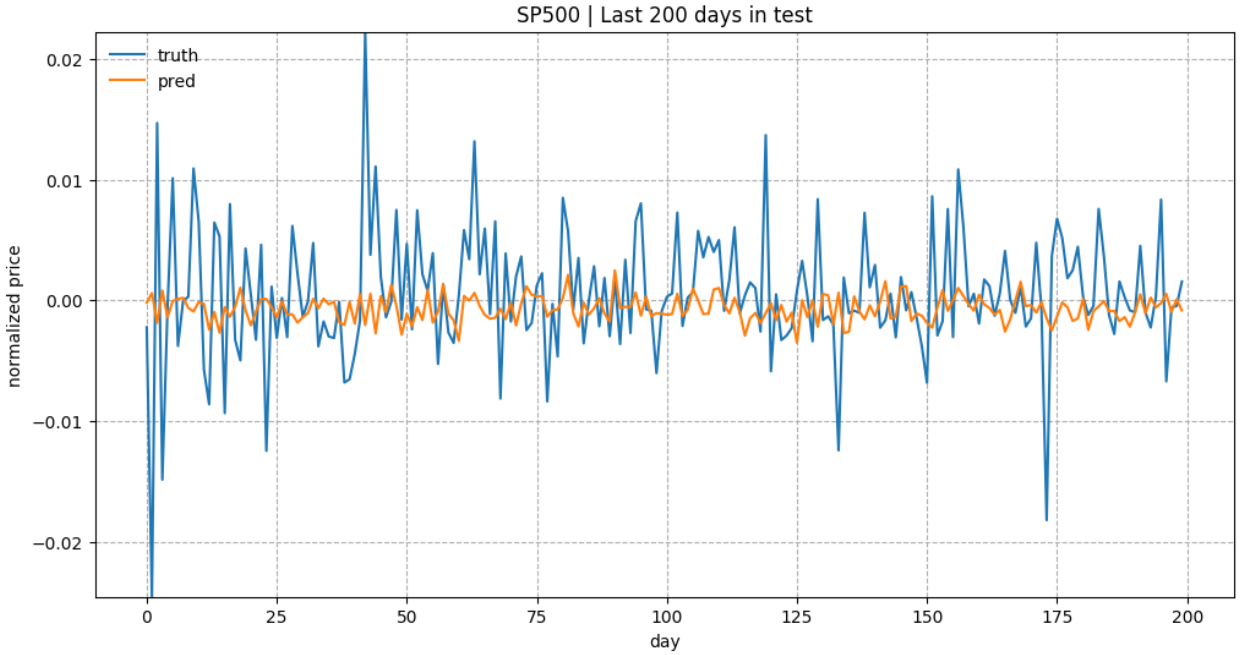


Рис. 17a. Результаты прогноза за последние 200 дней в тестовых данных.

Модель обучается с размером `input_size` = 1 и размером `lstm_size` = 32.

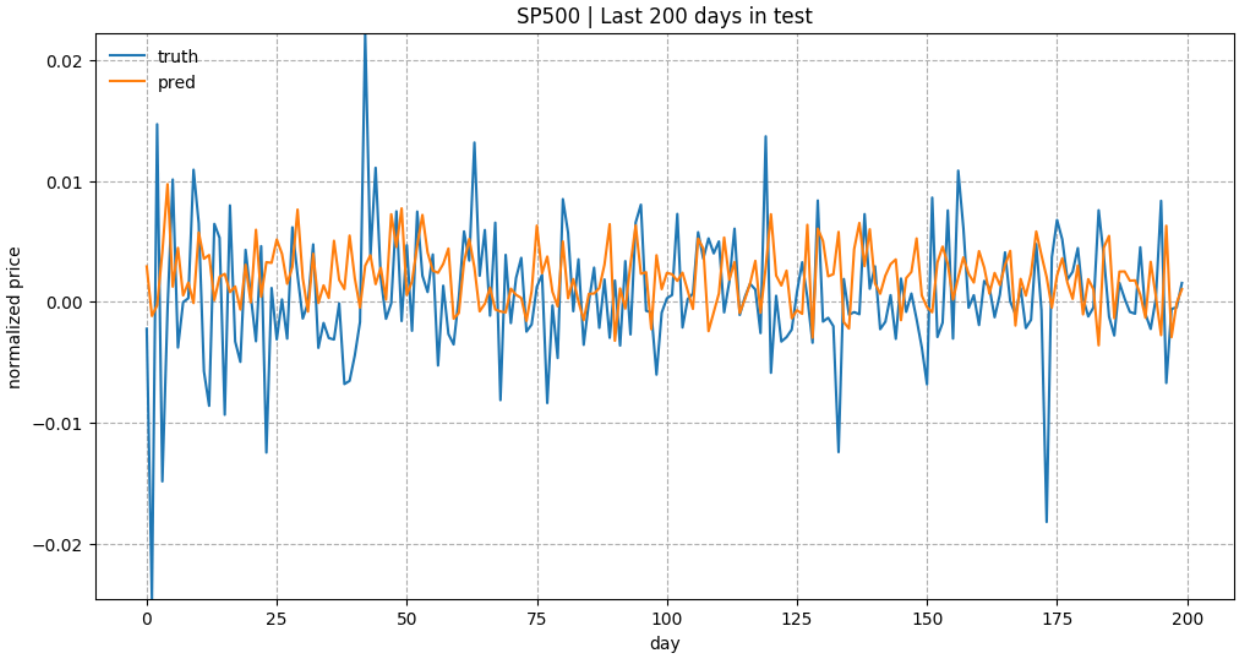


Рис. 17b. Результаты прогноза за последние 200 дней в тестовых данных.

Модель обучается с `input_size` = 1 и `lstm_size` = 128.

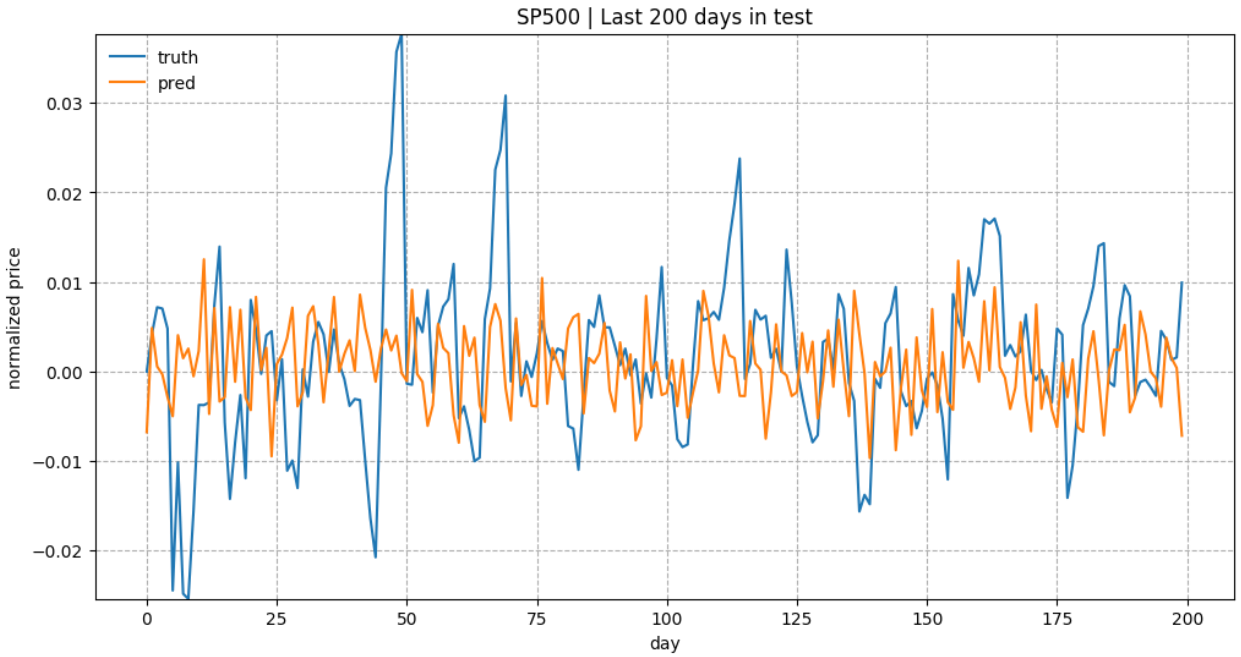


Рис. 17c. Результаты прогноза за последние 200 дней в тестовых данных. Модель обучается с помощью `input_size` = 5, `lstm_size` = 128 и `max_epoch` = 75 (вместо 50).

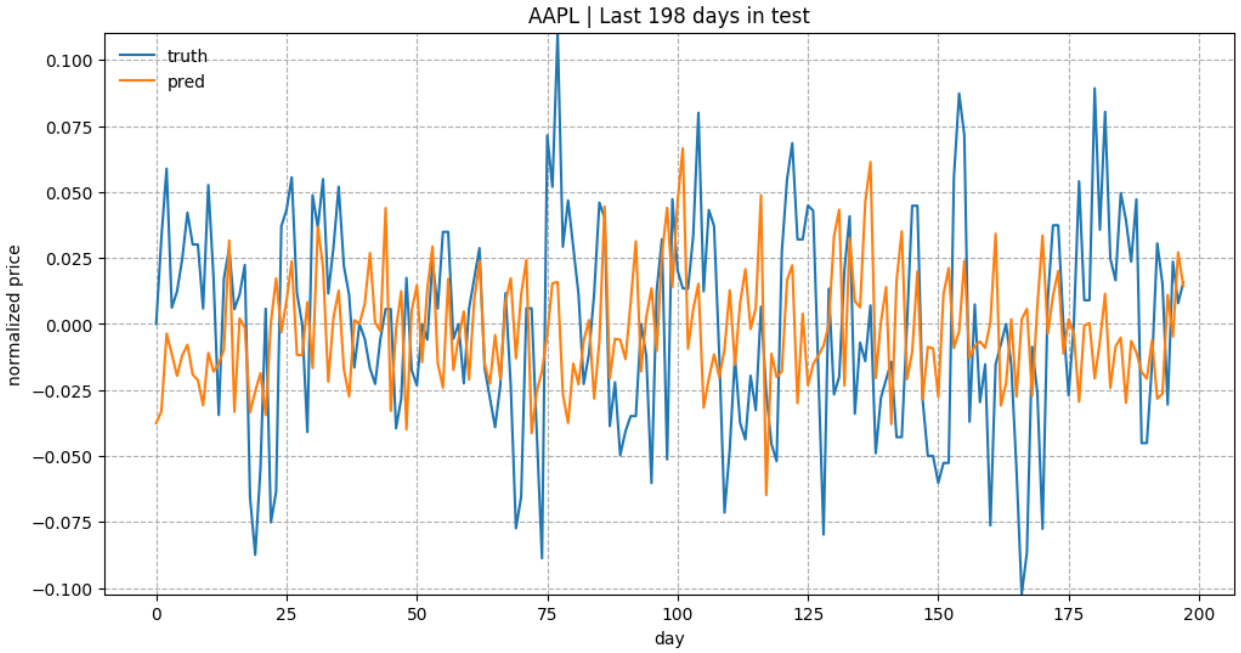
Во втором эксперименте используется следующая конфигурация (табл. 4):

`stock_count`	100
`input_size`	3
`embed_size`	3
`num_steps`	30
`lstm_size`	256
`num_layers`	1
`max_epoch`	50
`keep_prob`	0.8
`batch_size`	64
`init_learning_rate`	0.05

learning_rate_decay	0.99
init_epoch	5

Табл. 4. Эксперимент #2.

Для краткого обзора качества прогнозирования на рис. 18 представлены прогнозы для тестовых данных «AAPL», «GOOG» и «MSFT». Общие тенденции совпадают между истинными значениями и прогнозами. Рассматривая, как спроектирована задача прогнозирования, модель опирается на все исторические точки данных, чтобы прогнозировать только следующие 5 (input_size) дней. При малой величине input_size модели не нужно беспокоиться о долгосрочной кривой роста. Как только мы увеличим input_size, предсказание будет намного сложнее.



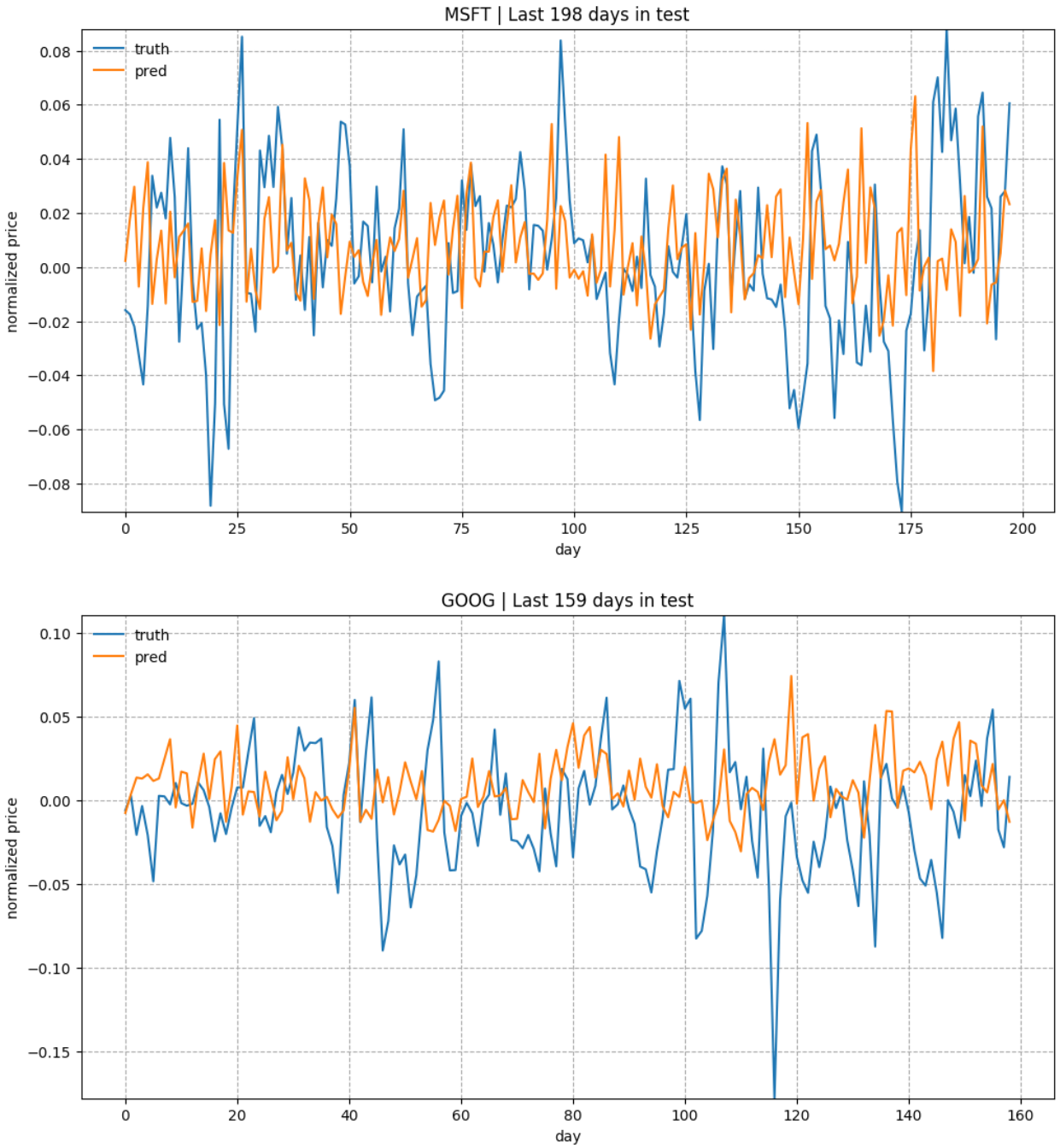


Рис. 18. Истинные и прогнозируемые цены акций AAPL, MSFT и GOOG на тестовом наборе данных. Цены нормализуются в скользящих окнах. Значения оси Y умножаются на 5 для лучшего сравнения между истинными и прогнозируемыми тенденциями.

3.2. Визуализация

Граф модели, определенный в коде, можно проверить визуально в **Tensorboard**, чтобы убедиться, что компоненты построены правильно. По существу это очень похоже на иллюстрацию архитектуры на рис. 16.

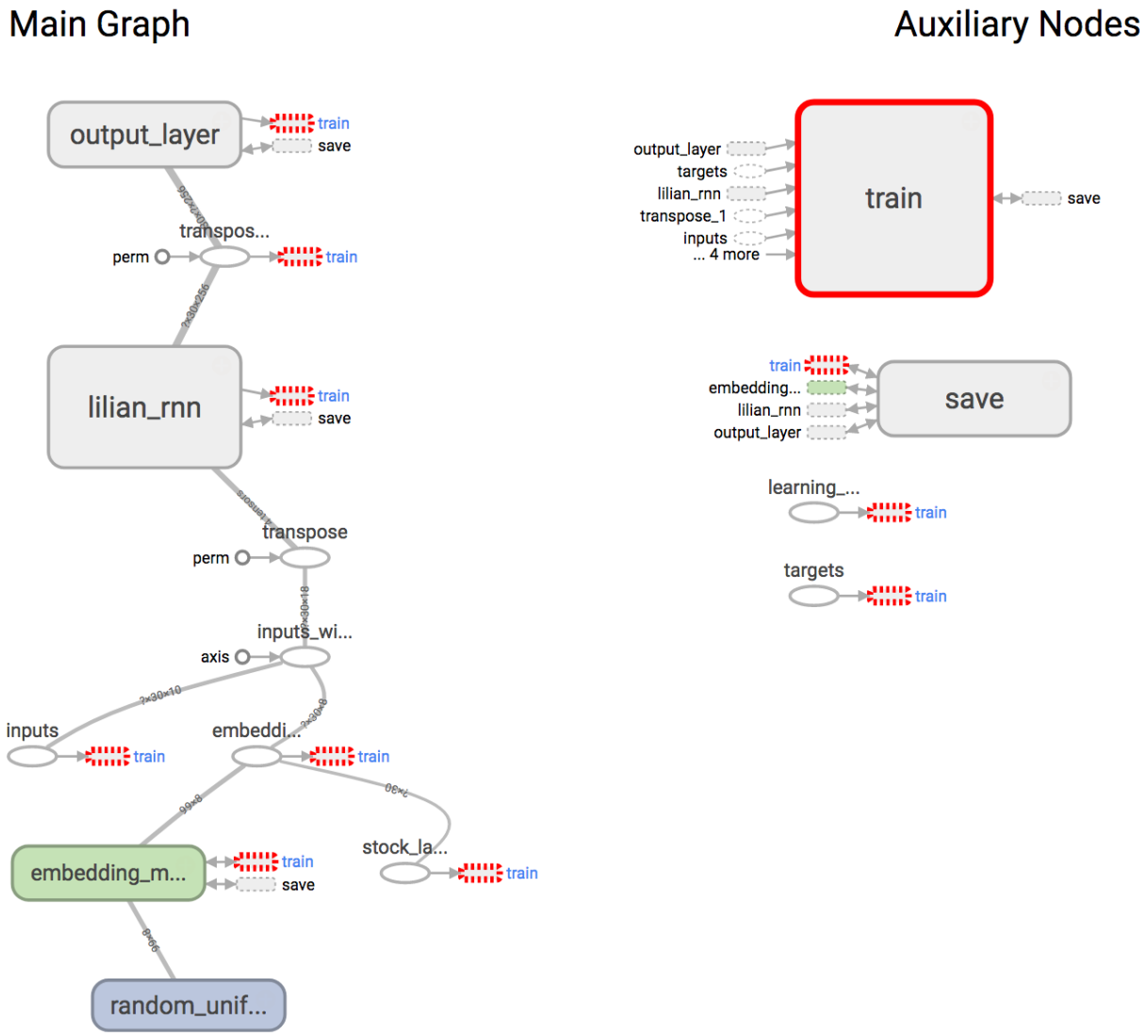


Рисунок 19. Визуализация графа модели в Tensorboard. Два модуля, train и save, были удалены с основного графа.

Помимо представления структуры графа или отслеживания переменных во времени, Tensorboard также поддерживает визуализацию

вложений [49]. Чтобы передавать значения вложений в Tensorboard, нам нужно добавить правильное отслеживание в журналах обучения.

Одним из распространенных методов визуализации кластеров в пространстве вложений является **t-SNE** [50], который хорошо поддерживается в Tensorboard. **t-SNE**, сокращенное для «**t-Distributed Stochastic Neighbor Embedding**» [51], является вариацией **Stochastic Neighbor Embedding** [52], но с измененной функцией затрат, которую легче оптимизировать.

1. Подобно SNE, t-SNE сначала преобразует высокоразмерные евклидовы расстояния между точками данных в условные вероятности, которые представляют сходства.
2. t-SNE определяет аналогичное распределение вероятности по точкам данных в низкоразмерном пространстве и минимизирует **расстояние Куллбака-Лейблера** [53] между двумя распределениями относительно местоположений точек в пространстве.

Настройка параметров, уровень сложности и скорости обучения (epsilon) в визуализации t-SNE может осуществляться произвольно [54].

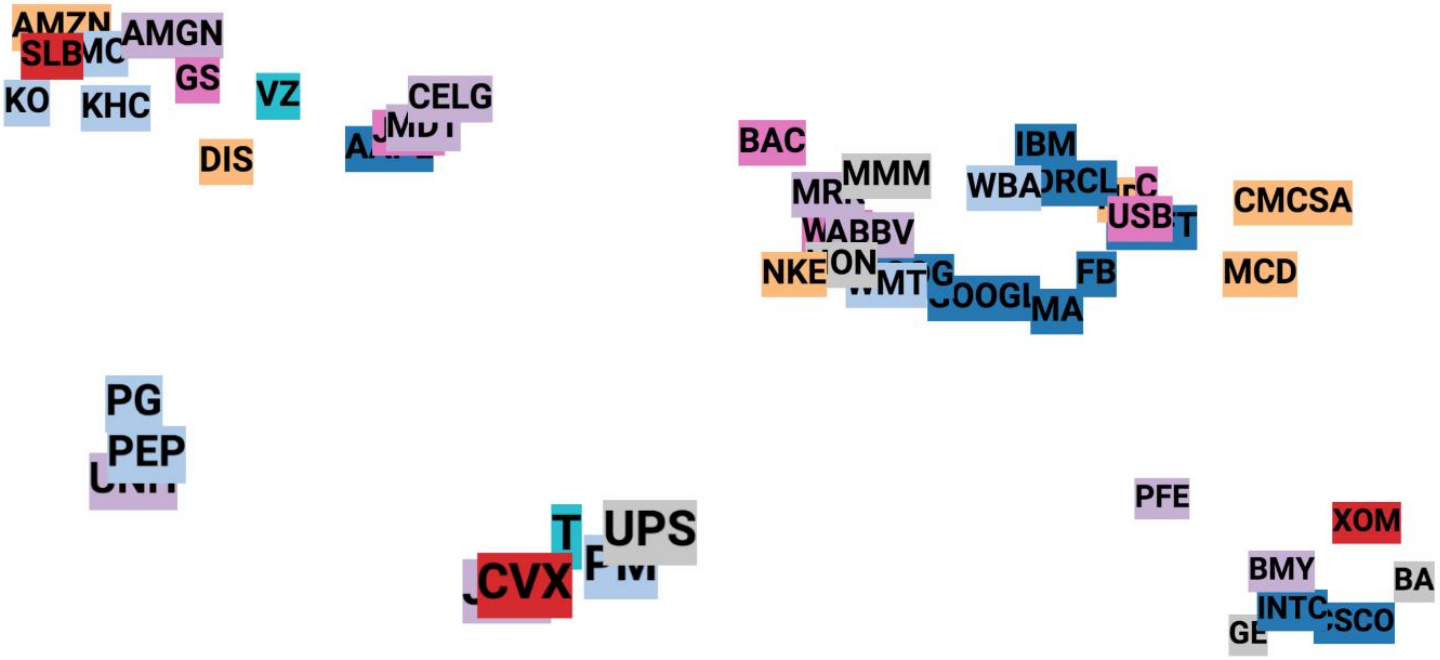


Рис. 20. Визуализация вложений акций с использованием t-SNE. Каждая метка окрашена в зависимости от отрасли промышленности. У нас есть 5 кластеров. GOOG, GOOGL и FB принадлежат к одному кластеру, в то время как AMZN и AAPL остаются в другом.

В пространстве вложений мы можем измерить сходство между двумя акциями, исследуя сходство между их векторами вложения. Например, GOOG в основном похож на GOOGL в изученных вложениях (см. Рис. 21).

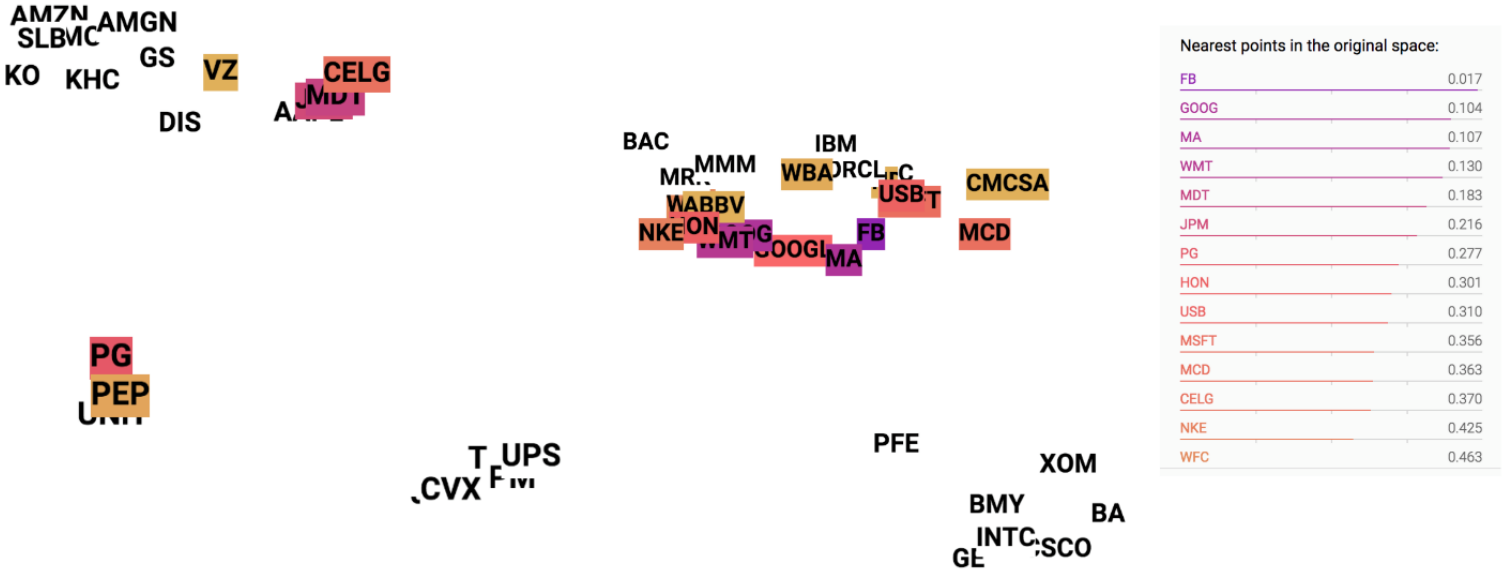


Рис. 21. «GOOG» выбран на графике визуализации вложений, а 20 лучших похожих соседей подсвечиваются цветами от наиболее темного до самого светлого, когда сходство уменьшается.

ВЫВОДЫ

Предсказательные уменьшаются и сглаживаются довольно часто по мере прохождения обучения. Поэтому необходимо домножать абсолютные значения на константу, чтобы сделать тенденцию более заметной как на рис. 18, поскольку нам необходимо знать, правильно ли прогнозируется направление движение цены: вверх или вниз. Тем не менее, должна быть причина для уменьшения значения предсказания. Теоретически вместо того, чтобы использовать простой MSE в качестве функции потери, мы можем принять другую, чтобы при неправильном предсказании направления штрафовать сильнее.

Функция потерь быстро уменьшается в самом начале, но она страдает от случайного взрыва (внезапный пик происходит, а затем

немедленно возвращается обратно). Скорее всего это связано и с её формой. Обновленная и более адекватная функция потерь может решить проблему.

ЗАКЛЮЧЕНИЕ

В данной работе был проведен анализ различных методов глубинного обучения, библиотек и инструментов, находящихся в открытом доступе для задачи прогнонозирования движения цен на финансовых рынках. Осуществлен поиск, предобработка и нормализация исторических ценовых данных S&P500 для обучения и тестирования модели, сами результаты визуализированы с помощью Tensorboard.

В процессе мы получили программный комплекс для прогнозирования направления движения цен, реализующий архитектуру рекуррентной нейронной сети (RNN), построенной на элементах долгой краткосрочной памяти (LSTM). Выбор сделан в пользу именно такой архитектуры, поскольку история движения цен представляет собой временной ряд и необходимо регулярно обращаться к ней и учитывать долгосрочный контекст, а RNN с LSTM для этого и предназначены.

Программный комплекс был реализован с помощью TensorFlow по ряду причин: многочисленное сообщество, обширный выбор документации, открытый исходный код, распределенное выполнение моделей, различные архитектурные оптимизации, визуализация во время выполнения (что упрощает отладку модели), портативность вплоть до того, что модели можно запускать на смартфонах с Android.

Существует множество способов улучшить результат: дизайн слоев и нейронов, разная инициализация и схемы активации, комбинирование с другими моделями (например, с интеллектуальным анализом текста) и т. д. Стоит упомянуть наличие целого класса микропроцессоров и сопроцессоров (ИИ-ускорителей)[55], которые можно применить для ускорения работы уже существующих моделей благодаря высочайшему уровню параллелизма. Чтение академических источников тоже может помочь глубоко изучить материал и идти в ногу с передовыми разработками.

Список литературы и источников

1. Голова и плечи (разворотная фигура) - http://berg.com.ua/chart-pattern/head-and-shoulders/
2. Кружка (фигура продолжения тренда) - http://berg.com.ua/chart-pattern/cup-with-handle/
3. Виды средних скользящих (SMA, EMA, WMA) - http://berg.com.ua/indicators-overlays/types-of-moving-averages/
4. Японские свечи - https://en.wikipedia.org/wiki/Candlestick_chart
5. Artificial Neural Network - https://en.wikipedia.org/wiki/Artificial_neural_network
6. Генетический алгоритм - https://en.wikipedia.org/wiki/Genetic_algorithm
7. Zhang, Y.; Wu, L. (2009). "Stock Market Prediction of S&P 500 via combination of improved BCO Approach and BP Neural Network". Expert

systems with applications. 36 (5): 8849–8854. doi:10.1016/j.eswa.2008.11.028.

8. Рекуррентная нейронная сеть - https://en.wikipedia.org/wiki/Recurrent_neural_network
9. Сеть с временной задержкой - https://en.wikipedia.org/wiki/Time_delay_neural_network
10. Thawornwong, S, Enke, D. Forecasting Stock Returns with Artificial Neural Networks, Chap. 3. In: Neural Networks in Business Forecasting, Editor: Zhang, G.P. IRM Press, 2004.
11. Robert P. Schumaker, Hsinchun Chen. Textual Analysis of Stock Market Prediction Using Breaking Financial News: The AZFinText System. Artificial Intelligence Lab, Department of Management Information Systems, The University of Arizona, 2006.
12. Mr. Raj Thakur, Miss. Sanchita Badkas, Miss. Meenakumari Pade, Mrs. Pallavi Khude. Stock market prediction using Neural Networks and sentiment analysis of News Articles. D.Y. Patil College of Engineering, Akurdi, Pune. 2017. DOI URL: http://dx.doi.org/10.21474/IJAR01/3749
13. Felix Ming Fai Wong, Zhenming Liu, Mung Chiang. Stock Market Prediction from WSJ: Text Mining via Sparse Matrix Factorization. Princeton University. arXiv:1406.7330v1 [cs.LG] 27 Jun 2014.
14. Jerry Chen, Aaron Chai, Madhav Goel, Donovan Lieu, Faazilah Mohamed, David Nahm, Bonnie Wu. Predicting Stock Prices from News Articles. Berkeley. December 11, 2015.
15. Xiao-Qian Sun, Hua-Wei Shen, Xue-Qi Cheng, Yuqing Zhang. Market Confidence Predicts Stock Price: Beyond Supply and Demand. University of Chinese Academy of Sciences, Beijing, China. 2016.

16. Mark Dunne. Stock Market Prediction. University College Cork. 2016.
17. Amin Hedayati Moghaddama, Moein Hedayati Moghaddamb, Morteza Esfandyari. Stock market index prediction using artificial neural network. Sharif University of Technology, College of Farabi, University of Tehran, University of Bojnord, Iran. 2016.
18. Phuong Dang Toan Nguyen. VOLATILITY PREDICTION OF STOCK PRICE USINGNEWS ARTICLES: A HYBRID APPROACH. NATIONAL UNIVERSITY OF SINGAPORE. 2014.
19. Bag of Words - https://en.wikipedia.org/wiki/Bag-of-words_model
20. Именованная группа - https://en.wikipedia.org/wiki/Noun_phrase
21. Именованная сущность - https://en.wikipedia.org/wiki/Named_entity
22. Метод опорных векторов - https://en.wikipedia.org/wiki/Support_vector_machine
23. Text Mining - https://en.wikipedia.org/wiki/Text_mining
24. Факторизация - https://en.wikipedia.org/wiki/Factorization
25. Коэффициент Шарпа - https://en.wikipedia.org/wiki/Sharpe_ratio
26. Причинность по Грейнджеру - https://en.wikipedia.org/wiki/Granger_causality
27. Авторегрессионная условная гетероскедастичность - https://en.wikipedia.org/wiki/Autoregressive_conditional_heteroskedasticity
28. Extract, transform, load - https://en.wikipedia.org/wiki/Extract,_transform,_load
29. Wang, Haohan, Bhiksha Raj, and Eric P. Xing. On the Origin of Deep Learning. arXiv preprint arXiv:1702.07800, 2017. - https://arxiv.org/pdf/1702.07800.pdf

30. Yann LeCun, Yoshua Bengio, Geoffrey Hinton. Deep learning. - http://pages.cs.wisc.edu/~dyer/cs540/handouts/deep-learning-nature2015.pdf
31. Understanding LSTM Networks. August 27, 2015 - http://colah.github.io/posts/2015-08-Understanding-LSTMs/
32. Andrej Karpathy. The Unreasonable Effectiveness of Recurrent Neural Networks. May 21, 2015. - http://karpathy.github.io/2015/05/21/rnn-effectiveness/
33. Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. arXiv:1406.1078v3 [cs.CL] 3 Sep 2014. https://arxiv.org/pdf/1406.1078.pdf
34. Greg Corrado. Computer, respond to this email. November 3, 2015. - https://ai.googleblog.com/2015/11/computer-respond-to-this-email.html
35. Метод главных компонент - https://en.wikipedia.org/wiki/Principal_component_analysis
36. G. E. Hinton, R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. DOI: 10.1126/science.1127647. 2006. - https://pdfs.semanticscholar.org/7d76/b71b700846901ac4ac119403aa737a285e36.pdf
37. Обучение с подкреплением - https://en.wikipedia.org/wiki/Reinforcement_learning
38. Silver, David, et al. Mastering the game of Go with deep neural networks and tree search. Nature 529.7587 (2016) - http://web.iitd.ac.in/~sumeet/Silver16.pdf
39. Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. arXiv:1406.2661v1 [stat.ML] 10 Jun 2014. - https://arxiv.org/pdf/1406.2661.pdf
40. Al Gharakhanian. Generative Adversarial Networks – Hot Topic in Machine Learning. Jan 2017. - https://www.kdnuggets.com/2017/01/generative-adversarial-networks-hot-topic-machine-learning.html
41. TensorFlow - https://www.tensorflow.org/

42. Delip Rao. The Unreasonable Popularity of TensorFlow. July 7, 2016. - http://deliprao.com/archives/168
43. S&P 500 (^GSPC) - https://finance.yahoo.com/quote/%5EGSPC/history?p=%5EGSPC&guccounter=1
44. TensorBoard: Visualizing Learning - https://www.tensorflow.org/programmers_guide/summaries_and_tensorboard
45. Backpropagation through time - https://en.wikipedia.org/wiki/Backpropagation_through_time
46. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, lya Sutskever, Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research 15 (2014). https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf
47. Epoch - http://www.fon.hum.uva.nl/praat/manual/epoch.html
48. Вложение - https://en.wikipedia.org/wiki/Embedding
49. Embeddings - https://www.tensorflow.org/programmers_guide/embedding
50. Laurens van der Maaten, Geoffrey Hinton. Visualizing Data using t-SNE. Journal of Machine Learning Research 9 (2008) 2579-2605. - http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf
51. t-distributed stochastic neighbor embedding - https://en.wikipedia.org/wiki/T-distributed_stochastic_neighbor_embedding
52. Geoffrey Hinton, Sam Roweis. Stochastic Neighbor Embedding. - http://www.cs.toronto.edu/~fritz/absps/sne.pdf
53. Расстояние Кульбака — Лейблера - https://en.wikipedia.org/wiki/Kullback%E2%80%93Leibler_divergence
54. How to Use t-SNE Effectively - https://distill.pub/2016/misread-tsne/
55. ИИ-ускоритель - https://en.wikipedia.org/wiki/AI_accelerator
56. Comparison of deep learning software - https://en.wikipedia.org/wiki/Comparison_of_deep_learning_software
57. Machine Learning Frameworks Comparison. Hello Paperspace. Nov. 11 2016. - https://blog.paperspace.com/which-ml-framework-should-i-use/

58. Abdelrahman Ahmed. Choosing a Machine Learning Framework in 2018. Feb. 9 2018. - https://agi.io/2018/02/09/survey-machine-learning-frameworks/